UNIVERSITÉ DE MONTRÉAL

A PROFILE-BASED ENERGY SAVING SCHEME FOR OBJECT TRACKING SENSOR NETWORKS

OSCAR GARCIA GARCIA
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AOÛT 2006

Your file Votre référence
ISBN: 978-0-494-19303-7
Our file Notre référence
ISBN: 978-0-494-19303-7

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

A PROFILE-BASED ENERGY SAVING SCHEME FOR OBJECT TRACKING SENSOR NETWORKS

présenté par : Oscar Garcia Garcia

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. CHAMBERLAND Steven, Ph.D., président

M. QUINTERO Alejandro, Doct., membre et directeur de recherche

M. PIERRE Samuel, Ph.D., membre et codirecteur de recherche

M. GALINIER Philippe, Doct., membre

ACKNOWLEDGEMENTS

I would like to thank Professors Alejandro Quintero and Samuel Pierre, my Master's advisor and co-advisor, respectively, for their feedback in the area of sensor networks and for their support, guidance and advice that helped me complete my thesis.

I would also thank Gabriel, Thérence, Khaled, Thamer, Rafik, Bchara and all my past and present colleagues at LARIM for the enriching discussions we have had.

Finally, I would like to thank my family, my parents Carmen Garcia and Laureano Garcia and my brother Ruben Garcia. They have stood by me in all my projects, providing support and confidence. They are a constant inspiration to me.

RÉSUMÉ

Les réseaux de senseurs sans fil sont l'une des technologies les plus prometteuses pour la connectivité des utilisateurs à des données du monde réel. Leurs composants, des nœuds fixes ou mobiles, ont des contraintes qui rendent le design des tels réseaux original par rapport aux réseaux ad hoc. Une de ces contraintes est celle de l'énergie; les nœuds fixes vont se trouver dans des environnements souvent d'accès difficile et les batteries ne seront pas en mesure d'être rechargées. Ceci posera des problèmes de durée de vie au réseau global. Une des principales applications de ces réseaux seront les réseaux dédiés au dépistage d'objets mobiles (*Object Tracking Sensor Networks*, en anglais). Plusieurs algorithmes ont été proposés dans la littérature pour gérer ces réseaux et les états de leurs nœuds d'une façon économe en énergie. La plupart de ces algorithmes sont basés sur la prédiction du mouvement.

L'idée fondamentale de notre approche consiste à utiliser l'information historique sur les déplacements des objets surveillés pour prédire les nœuds qui vont successivement héberger l'objet et ainsi activer seulement les nœuds qui sont nécessaires et pendant le temps nécessaire. La probabilité que l'objet suive son propre profil de mouvement n'est pas 100%, mais c'est aussi le cas pour la précision des méthodes de prédiction du mouvement. Les senseurs sont en outre placés dans une bonne position pour construire ces profils et apprendre le comportement de l'objet, ce qui peut ouvrir la porte à d'autres applications personnalisées aux utilisateurs qui se déplacent dans la zone surveillée, en minimisant l'énergie d'opération en même temps.

Les résultats obtenus montrent que notre algorithme obtient des performances intéressantes comparé à un algorithme bas à sur la prédiction, avec des économies de 20% même pour des prédictibilités de 60% selon la trajectoire de l'objet. En somme, la qualité des résultats obtenus lors de l'évaluation des performances met en lumière la capacité de notre protocole à gérer efficacement l'énergie dans les réseaux de senseurs de dépistage d'objets.

ABSTRACT

Wireless sensor networks are one of the most promising technologies to connect users to real world data. Their components, either static or mobile, have specific constraints that determine some especial considerations in the design compared to mobile ad hoc networks. One of this constraints is energy; the fixed nodes will find themselves in inaccessible places and batteries will be often non replenishable. The energy constraints will mean that the network will have difficulties to continue in operation if for the desired lifetime, since their deployment is an expensive task. One of the main applications of these networks will be object tracking. Several algorithms have been proposed in the literature so far to manage the energy in Object Tracking Sensor Networks (OTSNs). Most of these algorithms are based on movement prediction.

The basic idea in our approach is to utilize the information on the past history of t\movements of the object to predict the nodes that will host the object in the future. This will enable the network to activate just the nodes that need to be active for tracking the object and only during the time that the object will be near them, ideally. It is not 100% sure that the object will always follow its own behavior profile, but movement prediction is not sure either. Sensors are in a privileged position to build these profiles and learn the object's behavior and this functionality could set the path to new applications that are « user-aware » and at the same time minimize energy consumption.

The analytical and simulation results show that our algorithm has an interesting performance compared to prediction-based ones, reaching 20% savings even for 60% predictable profiles, depending on the type of trajectory the object follows. To sum up, the quality of the obtained results highlights the validity of our approach to manage the energy in an OTSN in an efficient way.

CONDENSÉ

Les réseaux de senseurs sans fil sont l'une des technologies les plus prometteuses pour la connectivité des utilisateurs à des données du monde réel. Leurs composants, des nœuds fixes ou mobiles, ont des contraintes qui rendent le design des tels réseaux original par rapport aux réseaux ad hoc. Une de ces contraintes est celle de l'énergie; les nœuds fixes vont se trouver souvent dans des environnements souvent à accès difficile et les batteries ne seront pas en mesure d'être rechargées. Ceci posera des problèmes de durée de vie au réseau global. Une des principales applications de ces réseaux est le dépistage d'objets mobiles.

Les réseaux de senseurs sans fil (RSF) ont des ressources limitées dû à des batteries faibles et la difficulté à accéder aux nœuds individuels, une fois déployés [1], [2]. Étant donné que l'énergie des nœuds peut être consommée dans des opérations de captation, calcul ou communication, la plupart des stratégies existantes dans la littérature essaient de réduire la consommation d'énergie dans les RSF en désactivant le plus de composants possibles du réseau et le plus longtemps possible. Dans le cas de l'application que l'on étudie dans cette mémoire, les réseaux de senseurs pour le dépistage d'objets mobiles ou *Object Tracking Sensor Networks* en anglais, idéalement les seuls nœuds qui devraient être actifs sont ceux qui suivent l'objet pendant sa trajectoire et seulement pendant le temps qu'ils le suivent. Parmi les approches proposés dans la littérature pour aborder ce problème, il y en a plusieurs qui s'appuient sur des techniques de prédiction du mouvement pour prédire en avance les nœuds qui devront être actifs et quand ils devraient être réveillés.

L'idée fondamentale de notre approche consiste à utiliser l'information historique sur les déplacements des objets surveillés pour prédire les nœuds qui vont successivement héberger l'objet et ainsi activer seulement les nœuds qui sont nécessaires et pendant le temps nécessaire. La probabilité que l'objet suive son propre profil de mouvement n'est pas 100%, mais c'est aussi le cas pour la précision des méthodes de prédiction du mouvement. Les senseurs sont en outre placés dans une bonne position pour construire

ces profils et apprendre le comportement de l'objet, ce qui peut ouvrir la porte à d'autres applications personnalisées aux utilisateurs qui se déplacent dans la zone surveillée, en minimisant l'énergie d'opération en même temps.

État de l'art en réseaux de senseurs sans fil pour le dépistage d'objets

Le problème de l'efficacité énergétique dans les RSF est présent en [1] et a été étudié d'une perspective générale en [8], [9]. Plusieurs stratégies spécifiques de minimisation de l'énergie ont été proposées en [3]-[7], [11].

Un bon nombre de travaux dans la littérature sur ce sujet proposent une architecture basée sur des grappes pour des raisons d'évolutivité et robustesse [3], [13]. Les têtes de grappe sont responsables de déterminer localement l'ensemble optimal de senseurs qui seront actifs pour minimiser l'énergie consommée dans le dépistage de l'objet. Les nœuds peuvent éteindre leurs unités radio à la condition que leurs mesures soient cohérentes avec les prédictions des têtes de grappe [3].

Deux stratégies principales ont été proposées dans la littérature pour achever des gains en économie d'énergie:

- Éteindre l'unité radio le plus longtemps possible, et
- Réduire la consommation d'énergie dans les composants de captation et calcul.

Dans un senseur, l'énergie consommée dans la transmission d'un paquet est autour du double de l'énergie consommé pour sa réception, et l'énergie consommée dans la réception d'un paquet est un ordre de magnitude plus grande que l'énergie consommée par exécution d'une instruction [4]. Ceci indique qu'une manière intuitive de réduire la consommation globale d'énergie est de réduire le nombre d'opérations gourmandes en consommation d'énergie aux dépenses d'une augmentation dans le nombre d'opérations qui consomment moins. C'est l'approche utilisée dans [4], qui présente l'algorithme PREMON. L'idée de base dans PREMON est d'échanger du calcul requis pour générer les prédictions par des économies en nombre de transmissions. La station de base qui est l'interface de l'utilisateur de l'application surveille les mesures des senseurs et génère des

modèles de prédiction, qui sont envoyés aux senseurs appropriés. Lors de la réception de ces modèles, les senseurs changent leur état de fonctionnement et envoient des messages de mise à jour à la station de base seulement lorsque leurs mesures divergent des prédictions plus qu'un seuil déterminé.

La réduction du temps d'activation de l'unité radio est aussi le but de l'approche proposée en [5], qui exploit la corrélation temporelle des mesures des senseurs. Deux senseurs voisins collaborent pour réduire l'énergie consommée par l'autre, en s'alternant pour activer leurs radios. En PREMON, les senseurs doivent garder leurs radios actives même lorsque les mesures sont prévisibles, pour répondre à des requêtes d'autres entités du réseau [4].

Une approche alternative pour réduire la consommation d'énergie au réseau est de diminuer les opérations de captation et de calcul. Ceci est l'approche proposée en [10] et [11]. L'algorithme proposé, PES, essaie de minimiser la fréquence d'échantillonnage (*sampling* en anglais) et le nombre de nœuds qui participent dans le dépistage. Le nœud qui suit l'objet a un moment donné (le nœud courant) déterminera un groupe de possibles localisations de l'objet mobile. Les prédictions sont calculées aussi au niveau des senseurs individuels en [11], et l'information recueillie au senseur n'est pas envoyée à sa tête de grappe si le mouvement de l'objet est cohérent avec les prédictions.

L'information des profils de comportement à long terme a déjà été utilisée en plusieurs contextes, comme les réseaux cellulaires [12] et les réseaux ad hoc [13]. Ces travaux se basent sur l'idée que le comportement des utilisateurs finaux dans la plupart d'environnements applicatifs est prédictible en grande partie. Nous utilisons cette supposition pour appliquer l'approche basée sur l'histoire et le profil au problème de la consommation de l'énergie dans les RSF.

Description du problème

L'application sous étude est le dépistage d'objets mobiles à l'aide d'un réseau de senseurs sans fil. Nous supposons une région de surveillance dont les frontières sont

connues par les applications qui vont récupérer les informations sur les objets mobiles. Les senseurs sont statiques.

Les senseurs dans les RSF comportent trois parties fonctionnelles: captation, calcul et communication.

Le sous- système de captation consiste en un groupe de senseurs qui font le lien entre le nœud et le monde physique. L'unité de calcul ou *Microcontroller Unit (MCU)* en anglais consiste en un microprocesseur ou microcontrôleur qui gère le fonctionnement des senseurs et exécute les protocoles de communication et les algorithmes de traitement des signaux. Il héberge aussi le système d'exploitation en temps réel des nœuds. L'unité de communication consiste en une radio de portée courte pour les communications sans fil. Un quatrième système est le sous-système d'approvisionnement d'énergie, qui héberge la batterie et fournit de l'énergie au reste du nœud [9].

Les modes d'opération disponibles dans le MCU sont [9]:

- Actif
- Inactif ou *idle* en anglais
- Endormi

Chacun des modes d'opération est caractérisé par une consommation d'énergie différente. La consommation d'énergie pour le mode "Inactif" est plus grande que celle pour le mode "Endormi". Les transitions entre les modes consomment aussi de l'énergie.

Nous faisons les hypothèses suivantes:

- Chaque senseur est une représentation logique d'un groupe de senseurs physiques qui décident ensemble sur les propriétés de l'objet mobile.
- Ces groupes sont formes au préalable avec un mécanisme de *clustering*.
- Les objets mobiles sont identifiables univoquement au moyen d'étiquettes électroniques et sont capables de transporter des données sur leurs propres mouvements et trajectoires.
- Les communications entre les nœuds et la station de base sont à plusieurs sauts.
- Les nœuds doivent être synchronisés avec l'application.

- L'interface entre l'application et le RSF est la station de base ou passerelle, qui reçoit de l'information sur les mouvements des objets qui a été envoyée par les têtes de grappe.
- Les senseurs connaissent l'identité du groupe ou *cluster* auquel ils appartiennent et ceux-ci sont identifies univoquement.

Les objets sont supposés de se déplacer toujours a l'intérieur de la zone étudiée, et d'être tout le temps a la portée d'au moins un nœud dans le réseau.

Toutes les opérations qui sont réalisées par le RSF consomment de l'énergie, soit la transmission, la réception ou le calcul.

L'application qui surveille les mouvements de l'objet a deux requis:

1. La durée de l'échantillonnage : les senseurs ont besoin de suivre les mouvements de l'objet pendant un certain temps, appelle « durée d'échantillonnage ». Pendant ce temps-ci, les unités de calcul et captation sont actifs mais les composants radio peuvent être éteints puisqu'il n'y aura pas de communication à la station de base [10].
2. La fréquence de rapport à la station de base détermine les moments ou les nouds devraient activer leurs composants radio pour communiquer avec la station de base. Les nœuds restants peuvent éteindre leurs unités radio.

La définition du problème reste alors la formulation d'une stratégie de minimisation de l'énergie qui réduit la consommation générale d'énergie dans les RSF de dépistage d'objets mobiles par rapport notamment a une stratégie basée sur la prédiction du mouvement et a un taux de perte de suivie de l'objet qui soit acceptable [10].

Le taux de perte d'objets est le ratio entre le nombre de rapports qui ont effectivement été envoyés et le nombre de rapports que l'application est supposée de recevoir du réseau [10]. Pour ces situations, de mécanismes de récupération sont prévus.

Solution proposée

Nous considérons qu'une stratégie basée sur des profils devrait faire mieux qu'un mécanisme base sur prédictions tant que les dépenses d'énergie causées par les situations non conformes au profil ne surpassent pas les gains.

Il est important de noter que la stratégie proposée n'est pas supposée de fonctionner mieux que l'approche basée sur prédictions pour toutes les applications et réseaux, mais seulement si la régularité des mouvements peut être observée et exploitée par les applications.

Ils existent certains compromis dans cette approche :

- Le temps de traitement associé aux requêtes et mises à jour de la base de données des profils peut être excessif pour acheminer les données temps-réel.
- Si l'objet mobile sort de son profil trop souvent, le profil doit être mis à jour et on doit exécuter un mécanisme de récupération. Le cout qui en résulte peut être plus grand que celui associé à la stratégie de prédiction du mouvement.

En conséquence, l'efficacité de la stratégie basée sur des profiles dépend de la précision de l'information qu'y est contenue et de la prédictibilité des objets mobiles.

Chaque objet mobile bâtit son profil au fur et à mesure qu'il passe a coté des nœuds du réseau. Le profil inclut les limites de temps pour les intervalles dans lesquels la journée est divisée, les identificateurs des têtes de grappe qui vont probablement héberger l'objet dans l'intervalle prochain et leurs probabilités respectives.

Chaque nœud qui détecte l'objet obtient une copie du profil de l'objet. Et pour chaque intervalle de temps, le nœud courant cherche dans le profil les nœuds qui sont supposés d'héberger l'objet lorsque l'application aura besoin de connaître la localisation de l'objet.

On s'attend donc à des économies en captation et calcul, étant donné que, si l'objet se déplace à l'intérieur de son profil, il n'y a pas besoin d'échantillonner son mouvement et pas besoin d'envoyer un rapport sur sa localisation à la station de base.

On va présenter des scenarios d'utilisation de notre algorithme.

- Scenario 1: l'objet est dans son profil lors de sa première détection.
- Scenario 2: l'objet est dehors son profil lors de la première détection.
- Scenario 3: a la fin de la période de rapport à l'application, l'objet n'a pas été détecté a nouveau et un mécanisme de récupération est déclenché.

Scenario 1

Au scenario 1, la première détection indique que l'objet mobile est dans son profil. Après que l'information du mouvement a été obtenue, l'objet est détecté encore une fois par un des nœuds qui sont actifs avant l'expiration d'un temps déterminé.

Vu que l'objet se trouve à l'intérieur de son profil, il n'y a pas d'envoi de rapport à la station de base, T secondes plus tard. Il n'y a pas besoin d'échantillonnage pendant X secondes non plus.

Néanmoins, le nœud courant va se réveiller T secondes plus tard, avec l'ensemble de nœuds le plus probables pour héberger les nœuds au prochain cycle de détection. Lorsqu'un des nœuds qui sont actifs est en train de capter les informations du mouvement de l'objet, il envoie un message d'ACK de retour aux nœuds courant. Ceci empêche le déclanchement d'un mécanisme de récupération.

Les étapes exécutées par l'algorithme dans ce scenario sont les suivantes:

- Le nœud courant détecte l'objet à l'intérieur de son profil.
- Il détermine l'ensemble de zones qui doivent être réveillées pour suivre l'objet au terme du cycle. Tous les nœuds dorment jusqu'au moment du rapport à la station de base. Aucun message n'est envoyé à la station de base car l'objet mobile est à l'intérieur de son profil.
- Ensuite, les nœuds qui ont été sélectionnés pour se réveiller reçoivent un message et ils passent à l'état actif. Ils essaient de trouver l'objet.
- Si un temps d'expiration passe et l'objet n'a pas été trouve, le nœud courant antérieur déclenche un processus de récupération (scenario 3).

Scenario 2

Dans ce scenario, la première détection indique que l'objet mobile est hors de son profil. Après avoir capturé l'information de son profil, l'objet est suivi encore une fois par un des nœuds qui sont actifs avant l'expiration du *timer*.

Le fait que l'objet est dehors son profil implique que l'échantillonnage est nécessaire et aussi la communication du rapport à la station de base.

Les étapes sont les mêmes que pour le scenario 1, mais le nœud qui a détecté l'objet doit échantillonner son mouvement pendant X seconds. Le profil est mis à jour et transféré a l'objet mobile, Un message est envoyé a la station de base étant donné que l'objet est dehors sont profil.

Pour le temps qui reste jusqu'au moment du rapport, tous les nœuds dorment, y inclus le nœud courant.

Les nœuds qu'on prévoit vont héberger l'objet vont recevoir un message pour se réveiller et l'objet qui va détecter l'objet va envoyer un message de confirmation au nœud courant.

Scenario 3

Dans une première étape, les nœuds voisins de nœuds qui ont été réveillés pour être dans le profil sont actifs. S'ils ne réussissent pas à trouver l'objet avant un certain temps d'expiration, le nœud courant antérieur va déclencher un processus de recherche global au réseau, qui est sur de pouvoir détecter l'objet selon nos suppositions.

Simulation et résultats

La simulation a été réalisée dans une plateforme PC-Windows à l'aide du logiciel de simulation Qualnet. On a utilisé un réseau de 30 nœuds et une station de base, plus une station mobile, et des trajectoires de l'objet plus ou moins prédictibles d'entre 30 et 80 nœuds. On a supposé un profil préexistant dans les nœuds. On a utilisé les caractéristiques de consommation d'énergie des nœuds du projet WINS. La portée de la radio des nœuds est de 15 m.

Les tableaux suivants nous aident à évaluer la validité de notre stratégie.

Résultats de simulation avec 1 nœud présent au profil

	Consommation Énergétique	Taux de pertes	Temps d'activation	Transmissions et Réceptions
60% probabilité cumulée	121%	160%	80%	126%
100% probabilité cumulée	76%	123%	65%	106%

Résultats de simulation avec 3 nœuds présents au profil

	Consommation Énergétique	Taux de pertes	Temps d'activation	Transmissions et Réceptions
60% probabilité cumulée	81%	168%	36%	156%
100% probabilité cumulée	68%	127%	35%	85%

Les résultats montrent une performance relative de GPBA par rapport à PES qui est intéressante. La consommation d'énergie est amélioré de 20-30% lorsqu'on a 3 nœuds dans le profil, et seulement 20% lorsqu'on a un nœud qui est présent un 100% du temps dans la trajectoire de l'objet. La performance de PES est meilleure seulement si un seul nœud est présent au profil avec un 60% de probabilité d'héberger l'objet. Ces résultats montrent que l'algorithme peut être avantageux si l'on considère la consommation additionnelle d'énergie pour l'étiquette électronique de l'objet et ses transmissions aux nœuds dans le réseau. La consommation d'énergie en calcul n'a pas été considérée non plus.

Le taux de pertes est moins avantageux dans notre algorithme que dans PES. Lorsque le profil peut expliquer le mouvement de l'objet seulement un 60% du temps, le nombre de rapports en retard relatif au nombre de rapports qu'il aurait du se produire est moindre que dans le cas de PES. Néanmoins, le cas ou on l'obtient une qualité de service meilleur est aussi le cas pour lequel on obtient les meilleures économies de

consommation d'énergie, spécialement lorsque les nœuds sont présents dans la trajectoire de l'objet tout au long.

La réduction dans le temps de captation est la source principale d'économies dans GPBA. Les économies peuvent atteindre un 70% et leur importance est plus significative que l'augmentation dans le nombre de transmissions et réceptions. Le nombre de transmissions et réceptions est réduit encore lorsque le temps d'activation est plus important et ceci explique les économies plus grandes d'énergie pour le cas pour lequel il y a plus de nœuds qui sont présents dans le profil et plus fréquemment au long de la trajectoire de l'objet.

Les résultats obtenus montrent que notre algorithme obtient des performances intéressantes comparé à un algorithme bas à sur la prédiction, avec des économies de 20% même pour des prédictibilités de 60% selon la trajectoire de l'objet. En somme, la qualité des résultats obtenus lors de l'évaluation des performances met en lumière la capacité de notre protocole à gérer efficacement l'énergie dans les réseaux de senseurs de dépistage d'objets.

Des avenues de recherche intéressantes sont l'apprentissage automatique du comportement de l'objet par le réseau a l'aide de techniques comme les réseaux de Bayes ou de neurones, le développement d'un algorithme de sélection de nœuds dans le profil qu'il faut réveiller en prenant compte du coût de communication inter-nœuds et du coût de captation et l'extension de la stratégie proposée au dépistage de plusieurs objets.

INDEX

INDEX OF FIGURES

INDEX OF TABLES

INDEX OF APPENDIXES

LIST OF ACRONYMS AND ABREVIATIONS

ACK	ACKnowledgement
API	Application Programming Interface
GPBA	Global Profile Based Algorithm
IP	Internet Protocol
LPBA	Local Profile Based Algorithm
MAC	Medium Access Control
MACA	Multiple Access Collision Avoidance
MACA/PR	MACA with Piggyback Reservation
MANET	Mobile Ad hoc NETwork
OTSN	Object Tracking Sensor Network
PES	PrEdiction-based Scheme
QoS	Quality of Service
RAM	Random Access Memory
UDP	User Datagram Protocol
WINS	Wireless Integrated Sensor networks
WSN	Wireless Sensor Network
WSAN	Wireless Sensor and Actor Network

CHAPTER I

INTRODUCTION

Wireless Sensor Networks (WSNs) are limited in resources due to the scarce power batteries they feature and the difficult access to individual nodes, once they are deployed [1],[2]. Since nodes' energy can be spent in sensing, computation or communications, most existing strategies intend to reduce energy consumption in the network by switching off as many components as possible and as long as possible. In the case of the application we study in this work, Object Tracking Sensor Networks (OTSNs), ideally only the sensors that follow the object should be active and only during the time the object moves near them. Some of the existing approaches rely on prediction techniques performed by the cluster heads to know in advance the next clusters to be activated and when they should be woken up. Knowledge of the regular patterns of things, animals or people can be utilized to extend the lifetime of WSNs by reducing their energy consumption. Moreover, the network can be a valuable tool to construct this profile information due to its density and proximity to the phenomenon [1].

In turn, Wireless Sensor and Actor Networks (WSANs) contain mobile, more capable nodes, called actuators or actors [2], which collect the information from the sensors (assumed to be static) when passing near them. The actors or actuators have the resources to take decisions based on that information and act in consequence. In OTSNs, mobile objects could easily have some memory resources to store their own movement information and carry it through the network. They could then be seen as actuators that exchange information with the sensors in the network.

This work presents a possible architecture and a strategy to take advantage of the profile information in order to improve the performance of the network making use of the capabilities of these mobile objects.

This introductory chapter presents the basic concepts of OTSNs, and the elements of the problematic, followed by our research's objectives and finally the outline.

1.1 Definitions and basic concepts

Wireless sensor networks offer two advantages over traditional sensor networks that are already implemented: the nodes are deployed in a sheer number, thus approaching the studied phenomena much more than before, and they have radio transmission capabilities to inform final users who perform data requests.

Wireless Sensor and Actor Networks (WSANs) belong to the category of ad hoc networks, but some differences arise that prevent one from using the same algorithms that were developed for ad hoc. The number of nodes in a WSAN is much bigger; they are more densely deployed and more prone to failures; moreover, their topology changes more frequently than that of ad hoc networks. A major feature of WSANs is the fact that no unique ID is attributed to the individual nodes; actually, an address can be assigned to some of the nodes, called cluster heads. As a consequence, the broadcast communication paradigm is preferred over the point-to-point approach of ad hoc networks.

Given the unpredictable nature of the wireless links and the scarce power resources available to sensors, communications between nodes will be multi-hop. We must also take into account the fragility of sensors, their irreplaceable batteries and the difficult access to humans who could do the maintenance. The WSANs are self-configurable networks.

The goal of energy saving is paramount to extend the lifetime of these networks; this is the motivation for the various algorithms existing in the literature for energetic efficiency; however, there exists a tradeoff between energy efficiency and delay in the network. The delay or time elapsed between data acquisition by the sensors and their arrival to destination is a critical quality of service metric in WSANs. WSANs need delay guarantees, especially in the case of certain applications (security) where the

actuators' actions would no longer be valid if the event's information arrives too late for the phenomena to rest unchanged. Moreover, actuator's mobility can add a distinct value to these networks, due to the fact that it can allow the actuators to track the events and react to changes.

Object or location tracking can be defined as the set of mechanisms by which the location information of an object is updated in response to its mobility.
In order to be able to track the object, we must first be able to localize the sensor nodes. This problem is called localization and is linked to the tracking problem. If we also want to identify the tracked object or even classify it, an identification problem arises.

1.2 Aspects of the problem

WSNs and WSANs are intended to perform high-level tasks by interacting with the physical world. In order to do this, they take advantage of their deployment density and the capabilities of the actuator nodes. Wireless sensor networks present considerable challenges: sensors are often placed in hostile, inaccessible locations, with limited energy sources and communication is the most energy-consuming activity [1].

Several algorithms have been proposed in the literature to use movement's prediction to optimize resource consumption in sensor networks. However, to the best of our knowledge, no algorithm has been yet proposed to utilize the predictability of mobile objects' behavior in many applications.

The traditional approach of a centralized database consists in having the central server maintain the current state of all the sensors. This model is energy-inefficient primarily because in a majority of cases, the readings of a sensor can be predicted temporally (from its past history) or spatially (from the neighboring sensors). In this scheme, all sensors are always active and sensing, and reports are sent to the sink every T seconds by the nodes that have new information from the objects in their areas [24].

The PREMON scheme proposed in [25] claims that readings should only be sent to the monitoring station (sink) when they differ from the predictions by more than a predefined threshold. We obtain thus savings in transmission, at the cost of more

receptions at sensors (the prediction information from the sink), more computation (prediction calculations at the sink or at a cluster head) and some more transmissions (the predictions themselves, which change less frequently than the readings do). If the predictions are accurate enough, the number of transmissions from the sensors to the base stations gets greatly reduced.

The predictability increases with the degree of correlation among readings of sensors in close proximity and with the degree of correlation between recent history and the readings that a sensor is going to see in near future [25]. Not all phenomena fall into this category, but a good number of them do (temperature, motion, pressure). For these, generating a prediction model with a small margin of error in the predictions is not computationally too expensive. This small error is tolerable in sensor networks.

The existing prediction-based techniques try to predict the future direction of the object's movement from its history. Calculations have to be performed at predefined nodes in the network in order to predict those locations. A profile-based scheme should outperform those mechanisms, provided that the overhead caused by the irregular situations (not predicted by the profile) do not overweigh the gains. Assuming the tracked object's behavior is predictable enough and the network can learn it, the energetic performance of the tacking network can improve over the time.

It is important to note that the proposed scheme is not expected to work better than the prediction-based ones for all the types of networks and applications, but for those in which regularity in the movements can be observed and exploited by the application. An especially suitable situation for our strategy is those applications whose goal is to trigger alarms caused by irregular behaviors (not according to history or profiles). For these applications, the profile-based strategy should maximize lifetime, by minimizing energy consumption.

1.3 Research goals

The main goal of our research is to propose an energy-saving scheduling approach for nodes' states in an Object Tracking Sensor Network by using the information on the

past behavior of the tracked objects while respecting the constraints imposed by the application. More specifically, the goals are the following:

- To analyze the existing solutions concerning energy management in OTSNs.
- To propose an energy-saving scheduling scheme in WSNs which is based on the tracked objects' movements' profile.
- To evaluate the performance of the proposed algorithm(s) by means of simulations, comparing them to the present situation of energy management in OTSNs which are based on movement's prediction, in order to measure the contributions of this work.

1.4 Outline

The rest of the report is organized as follows. Chapter II discusses the state of the art regarding the strategies that have been proposed to improve the energy consumption in OTSNs. Chapter III presents our strategy based on profiles of the past history information for the mobile objects. In Chapter IV, the algorithm's implementation in a network simulator and the results are provided. Chapter V gives the conclusion and the future work.

CHAPTER II

OBJECT TRACKING SENSOR NETWORKS

Recent advances in micro-electromechanical system technology (MEMS), wireless networking and embedded processing have enabled a new generation of massive-scale sensor networks that will couple user's needs to real-time, precisely localized information on the physical world [1].

In a near future, roads, walls, fields and machines can be populated with tiny and cheap sensors that will monitor vehicular, human or animal activity for a wide range of applications, both commercial and military. New technical challenges emerge with such new developments; some of them are common to wire line networks, as scalability and efficient use of bandwidth [1]. Other challenges are unique constraints to wireless sensor networks (WSNs).

2.1 Technical challenges unique to WSNs

A typical sensor network consists of sensors equipped with a microprocessor and a small amount of memory to perform calculations and task scheduling. They also have sensing units, which can be microphones, video cameras, seismic sensors...each node communicates wirelessly with neighboring nodes which are within their radio range [1].

The information captured from the physical world will likely be redundant among different sensors, due to its high deployment density. This calls for the need of data fusion to optimize the network's resources.

The information collected by such networks describes a condition of the environment (temperature, presence, humidity, vibration, pressure) and requires advanced level query interfaces and search engines. Sensor networks extend the Internet into the physical world, thus resulting in expensive networks (currently) and create different types of traffic to the existing ones in the Internet. The user's commands will

be routed via a central treatment location, which we will call sink. The sensor network will route its information through gateways which can be connected to the IP core network. The gateway would forward commands to the appropriate sensors and do data aggregation. Data storage devices can be attached to the IP network to support user-initiated search functions [6]. Figure 2.1, inspired from figures in [1] and in [19], shows a possible model to achieve this integration in a habitat surveillance application that needs a transit network to connect to the Internet [6]. The gateways collect and may treat the information captured by the sensor networks.

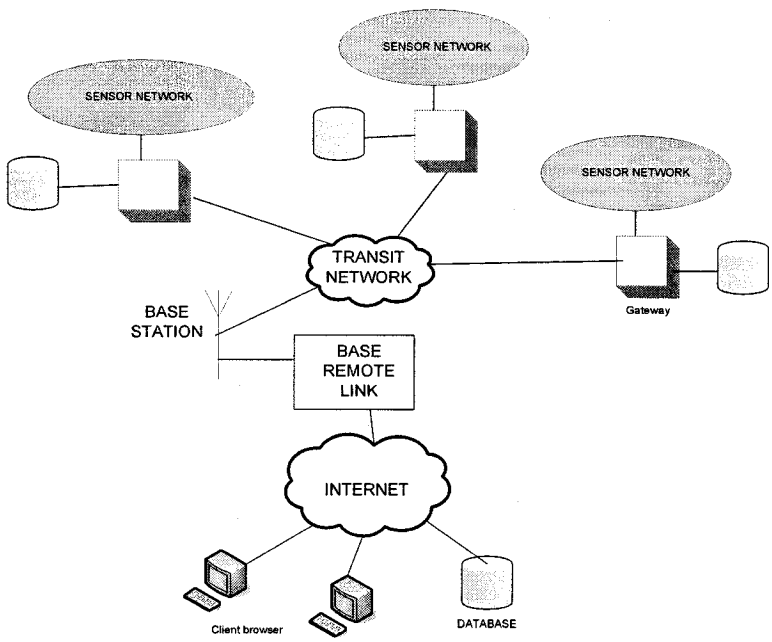


Figure 2.1 Integration of WSNs with the Internet

Wireless sensors hardware ranges from shoe-sized Sensoria WINS NG sensors to matchbox-sized Berkeley motes with an 8-bit microcontroller [17]. Their capabilities are summarized and compared in Table 2.1. For both models, the ratio of energy consumption for communication and computation is in the range of 1000 to 10000. Wireless communication will still be dominating in networked embedded systems in the near future.

Table 2.1 Comparison of several sensors' hardware

	WINS NG 2.0 Node	iPAQ with 802.11 and A/D Cards	Berkeley Mica Mote	Smart Dust*
Parts cost (quantity 1000+)	$100s	$100s	$10s	<1$
Size (cm3)	5300	600	40	.002
Weight (g)	5400	350	70	.002
Battery capacity (kJ)	300	35	15	(Less)
Memory	32 MB RAM, 32 MB flash	64 MB RAM, 32 MB flash	4 KB RAM, 128 KB flash	(Less)
Operating system	Linux	WinCE or Linux	TinyOS	(smaller)
Processing capability	400 MIPS/1.4 GFLOPS	240 MIPS	4 MIPS	(Less)
Radio range	100 m	100 m	30 m	(Shorter)

* Smart Dust is not yet fully operational, numbers are goals or estimations

Minimizing energy consumption in communication (by reducing amount or range of communication) is a design goal towards maximizing lifetime.

The challenges faced in designing sensor network systems and applications include [17], [19]:

1. Limited hardware: limited energy supply and bandwidth, processing, storage and communication capabilities in each node.
2. Limited support for networking: dynamic topology, mobile and unreliable connectivity. No universal routing protocols or central registry services. Each node acts as a router and an application host.
3. Limited support for software development: the software architecture must be designed in close cooperation with the information processing architecture because of the coupling between the applications and the system layers.
4. Scalability: the number of nodes can be several orders of magnitude higher than in an ad hoc network. Algorithms must be scalable.
5. Deployment density. Close nodes can collaborate to filter information previous to transmission of data to the application.

6. Fault tolerance and self-configuration: network topology evolves over time and applications need those characteristics.
7. Data-centric querying is different from the traditional address-centric one. Individual sensors are not aware of the total size of the network and do not have unique addresses. Paradigms as Directed Diffusion [14] impose a data-centered view of the network. The information is organized in <attribute, value> pairs. Nodes request data by disseminating interests for data throughout the network, and only data matching the criterion is relayed back to the node.

Even with these limitations and challenges, sensor networks present advantages to implement an application [19]:

1. Improvement of signal-to-noise ratio (SNR) for reduced distances from nodes to sources.
2. Detection advantage: each sensor has a finite sensing range, limited by the noise floor at the sensor. When a target is inside the sensing range of a sensor, increasing sensor density improves SNR, overcoming ambient environment factors which would interfere with the observation of the phenomenon.
3. Coverage of a large area is possible and can be shaped as needed to overcome obstacles. It can also be gradually extended by adding sensors in the zone of interest.
4. Multi-hop communications enable increased energy efficiency and in-network data aggregation of information from other nodes.
5. Improved robustness and scalability of the network as a whole against individual nodes or link failures, due to redundancy. This advantage calls for decentralized algorithms.

Sensor networks are designed to perform high-level information processing tasks, such as detection, tracking or classification [17]. They can detect false alarms or misses, classify errors or track quality, with well defined performance metrics.

Some sample sensor networks applications include [19] military applications (monitoring of forces, targeting, nuclear, biological and chemical attack detection), environmental applications (habitat monitoring, animal tracking, forest-fire detection, disaster relief, precision farming), health applications (patients' remote supervision, glucose-level monitoring for diabetics and even the development of an artificial retina for visually impaired individuals).

2.2 Wireless sensors and actors networks

The applications for WSNs become more interesting when nodes are mobile and can interact with the environment they are sensing. WSANs can be regarded as MANETs (Mobile Ad Hoc Networks) that have the following main features [3]:

- Dynamic topologies which change randomly, favoring the constitution of multi-hop links between every pair of nodes.
- Constrained bandwidth and wireless links with variable capacity.
- Energy-constrained operation.
- Limited physical security.

One of the challenges of this type of networks is routing. How can we find a route to the destination with a dynamic topology, non-reliable individual nodes and difficult to access nodes?

Many of the routing algorithms proposed for ad-hoc networks are not adequate for WSNs because they need unique addresses for nodes and continuous end-to-end updates. For instance, the Directed Diffussion (DDi) [14] algorithm does not specify a specific address but a geographic zone, and only needs the nodes to know their nearest neighbors.

Knowledge of neighbors leads usually to cluster-based algorithms. Clusters present certain advantages:

- Improved robustness and security.
- Simplified localization, routing and addressing.
- Reduction in energy consumption.

- Less memory requirements.
- Reduced delay in the control loop, since the actuators are usually close to the clusters or inside them.

In WSANs, sensors capture the information on events and transform it into electrical signals. These data are then sent to actuators, which are responsible for executing actions in consequence. It is often the case that one actuator only is not enough to perform a task and coordination with other actuators is necessary. New challenges have appeared regarding the requirements of communications between sensors and actuators in WSANs [3]:

- Nodes diversity: actuators are rich in resources.
- Required real time: we need little delay in the communication between sensors and actuators in order for these to perform an action which is still valid at the time of the data reception.
- Deployment: there are fewer actuators than sensors, since they have more resources. They have to cover all the area but at the same time, the communication must be possible between them.
- Coordination: a great part of the interactions is done between actuators and sensors, not just between sensors and treatment centre as in WSNs. This calls for the necessity of a local coordination mechanism between both types of nodes.
- Mobility: in WSANs the nodes are considered to be mobile, wheras in WSNs nodes are usually static.

Figure 2.2 shows the typical architecture of a WSAN. The treatment centre (also called "sink") is normally outside of the "sensors and actors field".

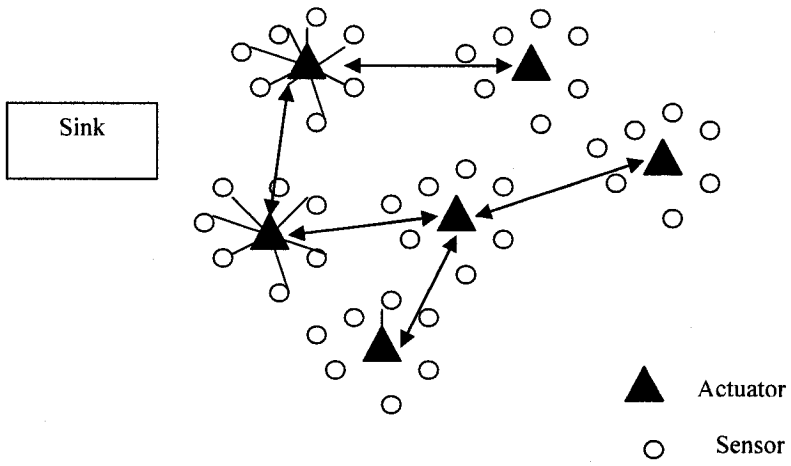


Figure 2.2 Architecture of a WSAN

Sensors can communicate their data either to the actuators or to the sink, which will ask the actuators to perform certain tasks. The first architecture is called « automatic » and the second one « semi-automatic » [3]. Table 2.2 shows a comparison between these two architectures

Table 2.2 Comparison of automatic and semi-automatic architectures in WSANs

Parameter	Architectures WSANs	
	Automatic	Semi-automatic
Delay	Low (sensors-actuators path)	High (sensors-sink-actuators)
Energy consumption	Low (only sensors in the area of the event)	High (also forwarding nodes)
Lifetime	Longer (congestion near the actuators which can move)	Shorter (congestion near the treatment centre, even with aggregation)
Algorithms	More complex (coordination between actuators)	More simplistic (the sink makes the coordination)

A model for the design of algorithms which schedules the sleeping time of nodes to save energy is proposed in [13]. The authors use random and deterministic sleep cycles and show that there is an energy reduction threshold which cannot be surpassed with this technique.

In some applications we find integrated sensors/actuators instead of actuators [17]. The sensing capacities of these actuators are limited and they will treat the information

from their sensors as well as those coming from the sensor nodes in the near area. The utilization of these nodes does not change the principles of the WSANs architectures.

2.2.1 Coordination between sensors and actuators

The communication between sensors and actuators is necessary to take decisions on the environment from the information taken by the sensors.

When designing coordination protocols for sensors and actors, two requirements must be taken into consideration:

- Low delays according to the application's requirements. There are not enough propositions in the literature to provide real-time communication in WSANs [17].
- Energy efficiency in communications between sensors and actuators, due to the scarce energetic sources of sensors.

Two aspects are to define when specifying a communication protocol for WSANs:

- Which sensors can communicate with which actuators, and
- How this communication takes place.

For the second aspect, two paradigms have been used in WSNs: single-hop and multi-hop. The election of one or the other depends on how the actuators are deployed.

In WSNs, the single-hop method is inefficient given the long distances between sensors and the sink. However, this is not always the case in WSANs, since the sensors are not far away from the actuators. In fact, if the actuator is in the centre of the events' zone, single-hop is advantageous, whereas if it is on the border or outside of the monitored region, multi-hop yields better energy consumption due to the smaller transmission distance.

Two paradigms are associated to the first problem: the affectation of actuators to sensors [3]:

- *Single Actor (SA).*
- *Multiple Actor (MA).*

Table 2.3 shows their advantages and downsides.

Table 2.3 Comparison between *Single Actor* and *Multiple Actor* architectures

Parameter	Number of actuators that receive the data fom the monitored region	
	Single Actor	Multiple Actor
Advantage	Short delay sensor-actor; No actor-actor coordination needed	Sensors consume less energy since they communicate with close neighbors.
	Efficient for concurrent and frequent events and few actors	Better utilization of richer actors' resources.
Downsides	Higher energy consumption due to more complex coordination.	Negotiation between actors increases delay and workload

SA: sensors belonging to a given zone will have to coordinate among themselves to decide which actor they transmit their information to. The criteria to make this choice can be [3]:

- Minimization of the average distance between sensors and the actor, in order to minimize the delays and the energy consumption of the nodes that will do forwarding and consequently minimize the network's lifetime;
- Minimize the energy on the communication paths between sensors and the actor.
- Aptitude of the actor to perform the required task, in terms of energy and action range, in an autonomous manner without help from other actors.

SA has the advantage of the rapid reaction of the actor, which does not need to invest time in coordination with other actors. In case the single actor is not capable of performing the task, it sends an announce message to the other actors and chooses one to execute the action based in their responses [3].

The downside in this approach is the need for a coordination mechanism between the sensors in the zone, which may cause an increase in the consumed energy.

MA : sensor nodes may transmit their data to several actors near the monitored region. Each sensor could decide which actor to communicate with. There could be no coordination mechanism between the sensors. The energy consumption could increase as compared to SA. To overcome this problem, we group the sensors in clusters, and each one is associated to an actor, rather than allowing each individual sensor to choose any actor. These clusters could be configured with a minimal distance to minimize the transmission delay, or with the minimal energy routes between sensors and actors to

maximize the network lifetime. Sensors spend fewer resources to select the reference actor, since they have to coordinate with neighbor sensors only, thus generating a smaller traffic load and less energy consumption than SA.

Actors must communicate among themselves to decide which will perform action. Since actors are resource-rich compared to sensors, MA takes advantage of them better than SA.

Moreover, the actors must be all connected in a graph that covers the entire region and allows all the actors to communicate to perform the required actors [3].

Another advantage is important in the case of mobile actuators; in MA several actuators could compare the intensity of received data from different zones and decide to move towards the centre of the events region to react more efficiently.

The downside of MA is the coordination mechanism between actuators, which is negotiation instead of announcement in SA. In MA, no actuator has all the information on the event and several transmissions and retransmissions can be necessary between actuators to take a decision. This could cause a greater delay and more traffic load.
Figure 2.3 illustrates both types of architecture, SA and MA.

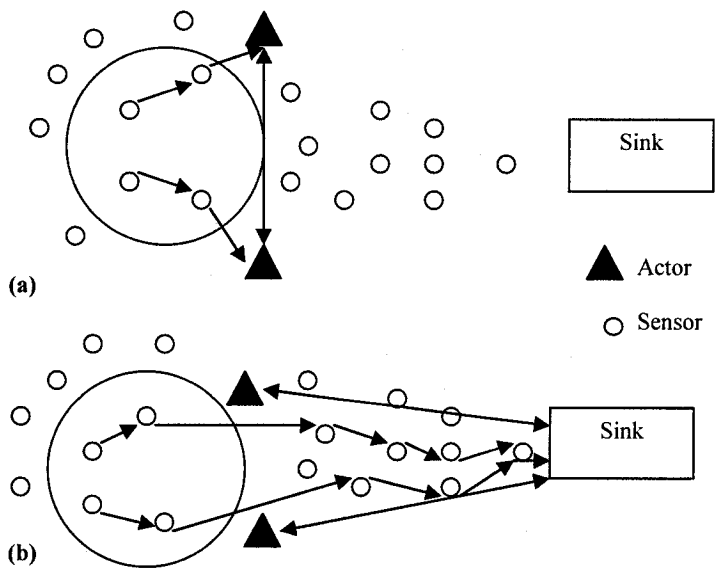


Figure 2.3 WSAN (a) with one actuator and (b) with several actuators

2.2.2 Coordination between actuators

Once the information of sensors arrives to the actuators, it's the actuators' responsibility to take decisions and react to the events. The actuators can communicate among them to do this. The utilization of several actuators can be necessary in the following situations [17] [20]:

- One single actuator cannot perform the required task or does not have the needed action range.
- Several actuators receive the same information and decide which one will react.
- Several actuators must react to several events.
- The event propagates itself towards the influence zone of another actuator. The interaction between the actuator which is currently in charge of the action and the actuator that is going to receive the information on the event is more rapid than the communication between the sensors of the new zone and the corresponding actuator.

Two reflection axes are presented in [17] to decide which actuators are going to participate and how to affect them the tasks:

- Single Actor Task (SAT) vs. Multi Actor Task (MAT).
- Centralized Decision (CD) or Distributed Decision (DD).

One model of affectation of tasks to "service providers" is introduced in [15]. The "service providers" are nodes in the network that offer services, defined as a series of actions. They are actuators. The model considers a sequence of tasks which arrive stochastically and the goal is optimizing the performance of the WSAN.

Metrics should be estimated for each service provider, but [15] does not develop them.

Let us assume that we have defined a task as being either SAT or MAT. The pertinence of using CD or DD depending on whether the network implements SA or MA showcases the factors indicated on Table 2.4.

Table 2.4 Single Actor Tasks in WSANs

Single Actor Task (SAT)		
MA (Multiple Actors)	Distributed Decision : Local negotiation between actuators to choose one.	Centralized Decision: direct transmission of data to the central actor, which selects the fittest.
SA (Single Actor)	If the delay requirement is very strict, the actor should decide and act for itself.	If delay is not critical or the actor cannot perform the task, it must communicate with the decision centre (CD) or with several actors (DD) to choose the fittest one.

In the case of multiple actors tasks (MAT), actors that receive the information (one or several, SA or MA) will absolutely need to determine the optimal number of actors to perform the task. This decision could be taken after communication with other actors in the DD case (by negotiation in the MA case or by announcement in SA), or with the decision centre in case of CD. Table 2.5 shows the main characteristics of the decision models, distributed and centralized.

Table 2.5 Centralized Decision and Distributed Decision in WSANs

Distributed Decision	Centralized Decision
Good response time thanks to the local coordination, regardless of the size of the network.	The decisions are taken in a centralized manner.
Need for algorithms to determine the fitness of an actuator to perform the task.	Resource savings in the actuators. Better for multiple actors tasks.
Need to define the structure of announcement messages or negotiation between actors.	If the centre is far away from the event, communication delays and need for location updates for mobile actors.

Generic challenges for inter-actor communication are:

- Development of algorithms that allow the actors to determine if the task should be performed by one or several actuators.
- Development of communication models between the actuators that include sensors as forwarding nodes for the situations in which the communication between actors is not feasible.
- Implementation of communication and coordination protocols that support real time operation.

2.2.3 Protocol architectures for WSANs

A protocol architecture for the WSAN is proposed in [3]. The structure contains three layers: communication layer, which enables the exchange of data between nodes in the network, the coordination layer which decides which actions the node should perform with the data and the management layer which controls de nodes regarding operation and survival. Figure 2.4 illustrates this architecture and the sub-layers proposed in [3].

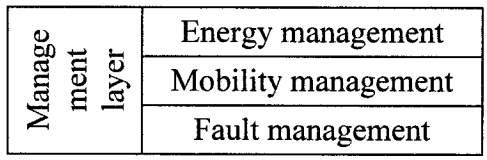


Figure 2.4 Architecture of a sensor and the associated sub-layers

The energy management layer informs the coordination layer about energy outages problems. The coordination layer can then decide not to take part in routing, or to command a mobile node to replenish its battery. The energy management layer maintains the actuators' location information, which must know the identity of neighbor nodes at every moment. The fault detection layer detects and fixes the hardware problems in the nodes and in that case it informs the coordination layer that the node in question will no longer participate in sensing, actuation or computation activities.

The communication layer (Figure 2.5) receives information coming from the coordination layer to communicate with other nodes in the network, and also receives communications from other nodes and sends the information to the coordination layer. This layer manages the medium access (MAC), the routes selection and the mechanism of end-to-end packets' transport [13].

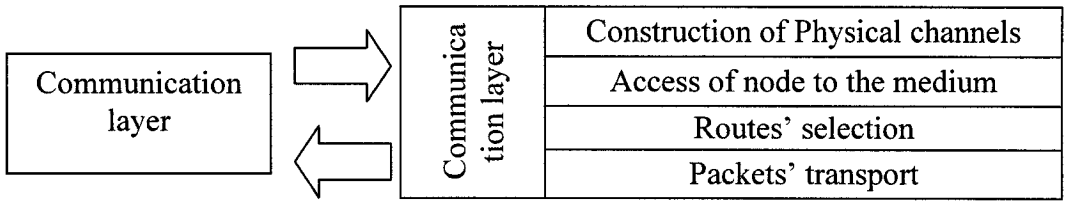


Figure 2.5 Communication layer in WSANs.

The transport protocols in WSANs have two main characteristics: reliability and reaction time. Regarding the communication between sensors and actuators, reliability is constrained to the event's properties (not individual messages) that allow the actuators to have the information required to take their decisions; on the contrary, reliability in communication between actuators refers to individual messages, which are all important in the exchanges that lead to actions [13].

2.2.4 Object Tracking Sensor Networks (OTSNs)

Object tracking is one of the most important applications in wireless sensor networks [21]. It can be a sole application by itself or part of a more general application. Object or location tracking can be defined as the set of mechanisms by which the location information of a moving object is updated as a response to an application's requirement. In order to be able to track the location of the moving object, the network must know its own nodes' position first: this is the localization problem. On the other hand, if the goal of the application is not only to track the movement of the object but also to classify or identify it, an identification problem appears.

Localization techniques can be classified in two categories:

- Neighbor-independent: every node knows its location without help from neighbor nodes. Examples are networks whose nodes are arranged in regular grids with positions fixed a priori, or GPS-equipped nodes.
- Neighbor-dependent: the calculation of the nodes' location depends on the cooperation of neighbor nodes. These techniques can be divided in centralized and distributed localization techniques. Centralized localization is

done in a central location and then transmitted to the sensor nodes. This means high communication costs, so it is not an adequate approach for WSNs. Distributed sensor network localization can be range-based or range-free. Range-based techniques depend on the distance or angle estimate, while range-free ones depend only on the contents of messages received from the other nodes.

Range-based techniques include the TOA (time of arrival), used in GPS systems, received signal strength and angle of arrival. Range-free techniques utilize the location information of anchor nodes, which are nodes with known positions.

The types of coordinate systems are:

- Absolute Coordinate System: an absolute reference for all nodes and objects. This method is used with GPS or several networks that are distant between them and have to be connected.
- Local Coordinate System: all nodes and objects in the network are referenced to a local reference.
- Relative Coordinate System: the locations of the nodes are relative to a reference, an anchor node. Different zones of a network can have different relative coordinates depending on the application requirements.

Localizing and tracking moving objects is an important capability of sensor networks in many applications. The main issues in such applications are [17] i) which nodes should sense, ii) which nodes have useful information and should communicate and iii) which nodes should receive the information and how often.

All these issues arise in a dynamically evolving environment. Object Tracking using sensor networks is an important problem for collaborative signal and information processing.

A tracking scenario is presented in [26]. As a target X moves across a sensor field, the following activities are initiated in the network:

1. Discovery: A node *a* detects X and starts tracking.

2. Query processing: one-run queries and long-running queries are both possible in the network. In the first case, the query are routed to the region of interest, the region around node *a*. Long-running queries mean that an end-user can always be aware of the objects' location.
3. Collaborative processing: Node *a* estimates the target location with help from other nodes. It can perform that estimation by triangulation, least-squares computation or, more generally, by a statistical method, such Bayesian Networks.
4. Communication: Node *a* hands off an initial estimation of the target location to the following nodes, *b*, *c* on the object's trajectory. A key issue here is how to choose the next node *b*, since a poor choice can cause a target missing and unnecessary communication overhead.
5. Reporting: one of the nodes along the trajectory may send back the data back to the querying node.

Another target Y could enter the field and the network would have to track both targets at the same time. The problem of properly assigning a measurement to a target is named association problem. Other problems raised in this scenario are:

- Target detection, localization, tracking, sensor tasking and control.
- Data naming, aggregation and routing.
- Data abstraction and query optimization.
- Time and location services, fault management, security.

Some commonly used measures of performance for the detection; tracking or classification tasks are [26]:

- Detection robustness (% missed and % false alarm).
- Detection spatial resolution (% counting error).
- Detection latency (event concurrence to query node notification).
- Classification robustness (% correct).
- Track continuity (% track loss).
- System survivability (network partition time; % track loss).

- Power efficiency (active lifetime, sleep lifetime, sleep efficiency). Network sleep efficiency is defined as the number of hours of target tracking versus the total number of hours in fully awake mode for the entire network. Percentage of track loss is the percentage of runs of a full scenario where a continuity of vehicle identity is not correctly maintained [26].

2.2.5 Energy management approaches in OTSNs

Sensor nodes have an extremely scarce energy budget. Battery replacement is not an option for networks with thousands of physically embedded nodes, so the nodes must have a lifetime on the order of months to years [24]. The power efficiency problem is introduced in [2] and studied from a general perspective in [24], [17]. Energy consumption is the most important factor that determines sensor node lifetime. Tracking involves inter-node collaboration functions such as Kalman filtering, Bayesian data fusion and coherent beam forming. This collaborative nature offers opportunities for energy management [24]. A well-structured and cross-layer design methodology is fundamental to maximize network lifetime; from the hardware platform to the application software and network protocols must be designed with energy-aware considerations in mind. A sensor node can be seen as consisting of four main functional units: 1) a microprocessor, 2) a communication subsystem with a short radio range, 3) a sensing subsystem wit sensors and actuators that link the node to the physical world, and 4) a power supply with a battery, that powers the rest of the node. An analysis of energy minimization opportunities at each of the units in a sensor node can be found in [24]:

- Microcontroller Unit: The microcontroller unit (MCU) controls the sensors and executes the communication protocols and signal processing algorithms. Several techniques have been proposed to estimate the power consumption of these processors. MCUs support several operating modes, including Active, Idle and Sleep mode, for power management purposes. Transitioning between operating modes also means a power and latency overhead.

- Radio: radios in general can operate in four modes: Transmit, Receive, Idle and Sleep. There is a significant amount of energy consumed when the radio switches from sleep mode to transmit a packet, due to the radio electronics. The Idle mode also consumes energy and it would be more efficient to shut down the radio rather than transitioning to Idle.
- Sensors: In general, active sensors such as imagers, mobile video cameras and sonar rangers, require much more power than passive ones, such as those measuring temperature, seismic, etc.

An important conclusion drawn in [24] is the fact that customized protocols which account for reception power consumption have to be developed. This is because conventional protocols have been assuming that the receive power is negligible.

Energy optimization in OTSNs can happen at the node level or network-wide [24]. Some guidelines for both approaches are the following:

- Node-level:
    - Power-aware computing: since state transitions have non-negligible energy consumption, the classical strategy of shutting down the node whenever possible is improved with dynamic voltage scaling: adapting the processor's supply voltage and operating frequency to just meet the current requirement.
    - Energy-aware software: at the core of the OS, the task scheduler can help enhance the network's lifetime. For example, scheduling the most important computations first protects the result of computations against premature failure of the node due to energy depletion. Energy scalability tradeoffs tracking precision, however.
    - Power management of radios. Startup power consumption and RF component electronics make this problem a non-trivial one. Algorithms to calculate efficient methods to schedule sleep time in the radio unit have been presented in the literature [12].

- Energy-aware packet forwarding: Intelligent radio hardware could identify packets that need to be redirected without having to process them, thus allowing the MCU to remain in sleep mode.
- Energy-aware wireless communication: optimizing power consumption in this component can yield high energy savings; modulation schemes could dynamically adapt to match the current traffic load. Distributing the algorithm's computation among several nodes would allow parallel execution and improve latency per computation, but it tradeoffs communication costs which would be necessary for internodes collaboration. The use of a good error control scheme could minimize as well the number of transmissions and reduce further energy consumption.

- Network-level:
  - Traffic distribution: to spread the load uniformly over the network is preferable over finding the minimum energy path for a multi-hop route and use continuously, for network lifetime considerations.
  - Topology management: In OTSNs, target tracking precision is enhanced by denser network deployments, but this tradeoffs network lifetime. This is because the radio energy consumption in Idle mode is not very different from that of Transmit or Receive modes, and changing to sleep mode changes the topology, so the Sleep mode transitions should be managed intelligently. An alternative approach is the utilization of a separate, low-power paging channel.

Several specific energy savings techniques are proposed in [3], [7], [9], [8], [11].

Nodes are most of the time sensing the environment and waiting for an event to happen. In this monitoring state, the radio could be turned off in all nodes, but as soon as an event is detected by a node, it would have to turn the radio on to initiate the multi-hop communication in order to report data to the sink. STEM utilizes this fact to keep the radio off until the data processing unit of the node decides that the information must be transmitted; the next hop is woken up by the first node and start listening to make the

communication possible [18]. Each node periodically turns its radio on to see if some node wants to communicate.

The use of a low-power paging channel that is always active assumes that its energy is much lower than the one used for regular communications, while STEM does effectively put the radio in sleep mode for some time. The wakeup protocol is run in a different frequency to that of data transmission, in order to avoid interferences. However, no consideration is given to the prediction of which nodes will have to be woken up or to the knowledge that the network might already have acquired.

Most works in the related literature propose a cluster based architecture for scalability and robustness [3], [23]. The cluster heads are responsible for locally determining the optimal set of sensors that will be activated so that energy spent on tracking is minimized. The nodes are allowed to turn off their radios as long as their measurements satisfy the predictions of the cluster-heads [3].

Two main strategies are used in the literature when proposing new power saving schemes:

- Turning off the radio as much as possible, and
- Reducing the consumption of the sensing and computation elements.

At a sensor, the energy consumed in transmitting a packet is around twice the energy consumed in receiving it, and the energy consumed in receiving a packet is an order of magnitude higher than the energy consumed per instruction-execution [7]. An intuitive way of reducing the global energy consumption is by cutting down the number of high-energy operations at the expense of an increase in the number of low-energy operations. This the approach utilized in [7], where the PREMON algorithm is presented. The basic idea in PREMON is to trade increased computation for generating prediction models for savings in number of transmissions. The base station (BS) monitors the readings of the sensors and generates prediction-models, which are sent to the appropriate sensors. On receiving these, the sensors change their working mode, sending an update to the BS only when their readings differ from the prediction by more than a threshold.

Reducing the time the radio is ON is also the goal of the Buddy protocol proposed in [8], by exploiting the temporal correlation in the readings of sensors. Two neighboring nodes collaborate to reduce each other's energy consumption, by taking turns to keep their radio ON. In PREMON, a monitored sensor node should keep its radio ON even when the readings are predictable, in order to be able to respond to requests from other entities in the network [7].

Avoiding unnecessary transmission is the goal of POTL (Prediction-based Object tracking Algorithm with load balance) algorithm [9]. The paper focuses on 1-D movement. Sensors only broadcast location information if it is different from those of predictions. The monitored region is divided in zones and each zone head is responsible for collecting movement information in the time slots in which it is active. The zone head rotates to minimize power consumption. Only the sensor with more power residue informs the zone head about the object's movement. The prediction of the next location is performed by the moving average method. A node sends a prediction report to the node that is expected to host the object. It this node does not find the object, a flooding recovery (all the nodes are woken up) is run. The proposed energy model achieves performance gains regarding to PREMON, since PREMON transmits information whenever the time slot arrives, and POTL only does so when the location measurements are different from the predictions. Nevertheless, these results come from an analytical study in a 1-D movement scenario and not from a simulation.

The missing probability is one of the performance metrics of OTSNs. A design factor that impacts this metric is the tracking interval, defined as the time length between two consecutive sensing points [4]. When the tracking interval becomes shorter, the resolution of the network is finer and the missing probability decreases. The Predictive Accuracy-based Tracking Energy Saving (PATES) is proposed in [4] to improve the power efficiency using the effect of prediction. According to the prediction received by the base station, a node is activated only when it is supposed to host the object in its detection area. The nodes report the movement history to the base station and this one performs the predictions and communicates the tracking interval to the nodes as well. If

during the tracking duration the measured movement is consistent with the prediction, no updates need to be sent to the base station. This saves long-distance transmission energy costs. In case that the object is missed because the prediction algorithm is not 100% accurate, two successive recovery stages are activated, waking up the neighboring nodes and then performing a flooding recovery if the first one did not succeed to track the object. The missing probability is derived as a function of the tracking interval and is fitted by a quadratic function in [4]. The authors finally obtain an optimal value for the tracking interval length with the assumption that the power consumption of the low power paging channel is negligible. They also observe that the miss probability is directly proportional to the target's speed, which means that a fast-moving target needs a shorter tracking interval. The work proposes a self-cognition capability in the network to optimize dynamically the tracking interval. It does not consider however the fact that the application could impose a tracking interval and the algorithm is not distributed, since the base station centralizes the prediction calculations and incurs in communication costs to inform the nodes.

An alternative way to reduce the energy consumption in the sensor network is cutting down sensing and computation. This is the approach proposed in [25]. The proposed algorithm, PES, tries to minimize the sampling frequency and the number of nodes involved in object tracking, while balancing off the overhead caused by missing the objects [25], where "nodes" means cluster heads. The node under which the mobile object is currently (the current node) will determine a group of possible location(s) for the mobile object. Predictions are calculated also at the individual sensors level in [25], so the information collected at the sensor node is not sent to its cluster head if the object's movement is consistent with predictions.

Information of long term history profiles has been used in several contexts, such as cellular networks [22] and ad hoc networks [23]. They are based on the assumption that the behavior of end-users in most application environments is predictable to a large extent. We use this assumption to apply the profile-based history approach to the energy consumption problem in WSNs.

The Prediction-Based Energy Saving scheme, called PES, minimizes the number of nodes participating in the tracking activities, and inactivates other nodes into sleeping mode [25]. The interface between the application and the sensor network is the sink or gateway, which receives the information on the objects' movements and makes it accessible to the final user. Figure 2.6 shows the general problem context. The nodes are deployed in a monitored region where the object is going to move. The sensors are static and the authors consider that the determination of the necessary movement information is a collaborative task, so the nodes the paper deals with are logical ones, actually an association of several physical nodes or cluster heads.

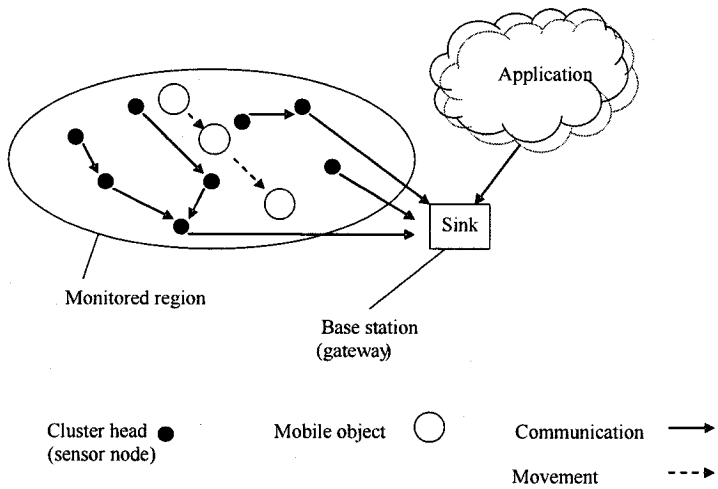


Figure 2.6 Object Tracking Sensor Network general architecture

The work also outlines basic schemes for energy saving in OTSNs:

- Naïve: all nodes are always active. The objects are always tracked (no object missing, assuming the object is always detectable by the network) and the information is sent to the base station or sink every T seconds (the time required by the application). This mode does not adjust the energy savings to the application requirements and offers always more energy consumption than the following methods. It is used as a bottom line reference in the analysis.

- Scheduled Monitoring: all the sensor nodes can turn to sleep and only wake up when it's time to sample the object and report the location information. The advantage is that the active time of every individual sensor is minimized. The disadvantage is the higher than necessary number of active nodes. Besides, time synchronization throughout the network is required.
- Continuous Monitoring: only one sensor node is activated, the one that hosts the object, but it remains active as long as the object stays in its detection area. The advantage is that only one node needs to be active to report the information to the sink.
- Ideal: the least-consuming approach would be one in which only one node would be active tracking the object and only during the sampling time. This means that the prediction should always be accurate. To determine which accuracy is needed to outperform the above presented methods is the research challenge addressed in [25].

In the PES scheme [25], the current node predicts the possible location(s) of the moving object and determines a group of sensor node(s), called target nodes, to help tracking the object after a period of sleeping. The authors show that PES reduces the energy consumption in sensor networks, compared with the naïve scheme. The PES scheme aims to reduce the time the current node is awake. If the object is not tracked by the target nodes after the sleeping period, an object missing occurs. To reduce the probability of object missing, other nodes besides the target ones are waken up. Moreover, a recovery mechanism is proposed if no acknowledge message arrives back to the current node within a certain time.

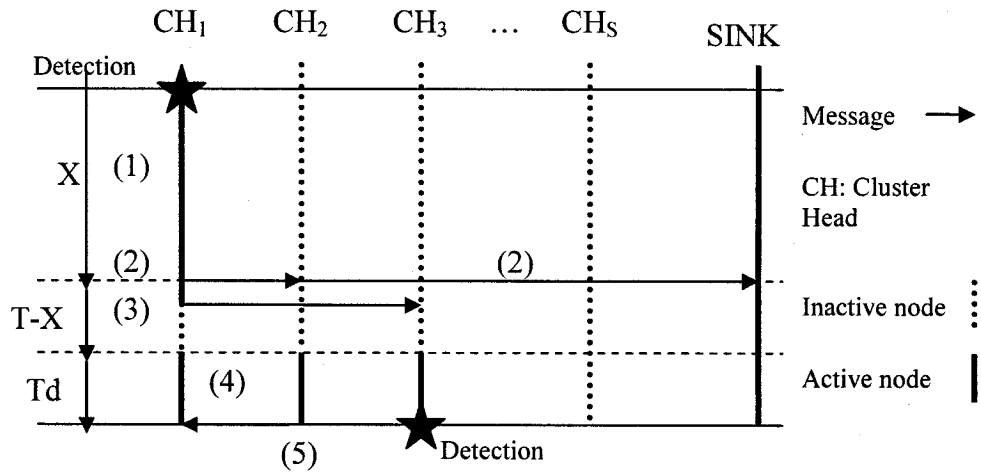


Figure 2.7 PES mechanism

The steps executed by the PES algorithm, as shown inFigure 2.7, are:

1. The current node that detected the object does the sampling during X seconds. The remaining nodes are sleeping.
2. After X seconds sensing, the current node predicts the object's movement characteristics for the next (T-X) seconds. It predicts the next destination node for the mobile object. It will wake up a set of so called 'target nodes'. The target nodes may be: only the predicted destination node (heuristic DESTINATION), or the nodes on the route to the destination node (heuristic ROUTE), or all the nodes neighboring the route (heuristic ALL_NBR). Wake up messages are sent to the target nodes with activation times. Upon reception of the wake-up messages, all the nodes wake up together at the activation time. Also at this step, a report of the readings is sent to the Base Station.
3. During (T-X) seconds, all nodes are sleeping.
4. After the (T-X) seconds interval, the target nodes wake up and start to track the object.
5. All the target nodes stay awake for the activation time T_d. If the prediction is accurate (accuracy is α), the object will be detected by one the target nodes and

sends an acknowledge message back to the current node. If the prediction is not accurate and the acknowledge message is not received, the recovery mechanism starts.

The recovery mechanism consists of two stages. They are executed when the current TL assumes a missing object after a certain timeout. If the first level of recovery succeeds in tracking the object, the second level (more costly) is skipped. β is the probability that the first recovery stage succeeds to track the object. A tradeoff exists in the recovery strategy; the more nodes that are to be activated, the higher the communication cost, but the lesser time it takes for the system to locate the object.

Two successive steps are taken, as stated in [25]:

1. The nodes neighboring the route from the current node to the destination are woken up by the TL. If no acknowledge is received within a certain time, the second stage of recovery is activated.
2. The second stage is a flooding recovery. All the sensor nodes in the network are woken up. This ensures 0% probability of missing the object, assuming it stays inside the monitored region.

The performance of the proposed algorithm is studied with CSIM simulator as to two metrics: total energy consumption and missing rate. The WINS nodes are used for the simulation and the topology model is a regular hexagon-shaped network. The number of objects used varies and is utilized to analyze the impact of network load, moving behavior and sensing operation. Some conclusions are:

- Continuous Monitoring is better than PES when the moving object changes direction and speed quickly. Otherwise, PES is preferable.
- Longer sampling duration increases the sensing operation cost but also increases the prediction accuracy. This is due to the shorter period of uncertainty in which the object is not sampled. The increment in sensing cost overwhelms the energy savings obtained by accurate predictions.

- When the sampling frequency is increased, the energy consumption is linearly increased, but the missing rate is dramatically decreased. The extra operation cost overwhelms the gains in prediction accuracy. Increasing the sampling accuracy is therefore not a good strategy to increase the prediction accuracy.
- As for future work, the authors mentioned looking into the moving patterns of the objects.

PES will be the main reference we will use to test our proposed algorithm, since it incorporates sensing and computing savings by using the movement prediction information. Nevertheless, their model does not consider the energy overhead for switching operating modes.

2.2.6 Use of Behavior Learning to optimize performance in Communication Networks

Wireless sensor networks will likely be embedded in the physical world and will provide huge amounts of data to be analyzed. How to deal with this amount of data before and after its sending to the end user is a research and operational challenge in itself. The data generated by the network can be seen as the examples that can be processed by learning algorithms. Supervised learning refers to learning from examples, in the form of input-output pairs. The result is a system that can estimate an unknown function without being programmed in advance, in such a way that it can predict the outputs for input values that are outside the training values [16]. A scalar magnitude represents the measured characteristic of the environment (may be arranged in vectors of scalars) and is associated to every node, which is a point in the space that represents the monitored region. A node generates a data set and applies the learning algorithm to obtain its estimate of the measured magnitude. In OTSNs, [16] the inputs are positions of sensor nodes and the outputs for each one are +1 (if one element of the set of objects is closer than a given threshold to the node) or 0 (otherwise). The threshold is the

estimation accuracy. Simulations with MATLAB are presented for several applications of WSNs.

Another technique for modeling and learning information from the environment in WSNs is based on a Bayesian classifier and is introduced in [5]. The work maps the problem of learning and utilizing contextual information to the problem of learning the parameters of a Bayes classifier, and then making inferences, respectively. They assume that nodes are "context-aware", this is, they are aware of their neighborhood and history. A comparison is made with Markov Random Fields, i.e., in temporal data, the influence of the past is completely summarized by the last observation, but in spatial data (images), the readings of a node depend only of its immediate neighbors. The sensors learn correlations with each available neighbor and weight them according to their degree of correlation. The sensor then chooses the neighbors that yield higher confidence. Once the correlations are learned they are reused over time, without need to continuously send predictions to a base station. This is called in-network processing and has been shown to be energy-efficient in communication. Applications of this Bayes-based approach are presented for detection of outliers, approximation of missing values and sampling.

Users' behavior and their profiles are utilized to simplify the implementation of Mobile Ad Hoc Networks (MANETs) and improve the performance of these networks in [23]. The authors propose some modifications in the network architecture to support a profile-based protocol for a specific application: the public bus transportation network. They utilize a database of profiles to store the information on users' behavior: mobility patterns, work schedules, likely communication requirements, resources more often used, etc. This database is manually installed on each bus. The architecture is composed of "profile agents" (software installed in every bus), which search the pertinent information in the profile database, retrieve new information for the profile and update it. The profile agent calculates the approximate location of all buses from the information that exists in the local profiles' database, the city map and the bus schedules.

Whenever a source wants to communicate with a destination, it calculates the shortest path to do so with the information provided by the profile agent. It is a likely route, but maybe not exact, given that several buses might be late and other nodes could have run out of resources. The source node transmits the data messages along the probable route, and each node that receives it without being the destination forwards it in case it is close enough to the likely route. Otherwise, it discards it. The node can know its distance relative to the probable route thanks to the profile agent and a floor value for the distance. A cache memory prevents more than one packet forwarding. If the calculated route is connected, the destination will receive the message. In case a bus is late or in advance, chances are that it will still be inside the zone where packets are forwarded. Simulations have been performed with GloMoSim using Edmonton transportation network and compares the profile-based algorithm with another algorithm which does not utilize profiles.

The use of the information on the cellular network users' behavior is the subject of [4]. In cellular networks, a mobile handset transmits its location to the system whenever it crosses a location area border. When the handset receives a call, the system pages it in all cells within the location area where it is known to be, according to the last location update. These two system activities (location update and search) consume bandwidth and resources, and several techniques have tried to reduce their impact on the global network.

The goal of location update is to reduce the location search cost ; if the mobile unit performed a location update each time it changes cell, the system would not need to search everywhere in the location area, but the location update would be higher. Therefore we find a tradeoff in the techniques designed to solve this problem.

Assuming that most users follow routines and they are expected to stay within a specific set of zones, if the network knows its location with a certain likelihood, the update cost could be reduced, but the location search cost increases if the user is not inside the first expected zone, but inside another expected zone. A data structure named User Mobility History (UMH), which contains the cells which will be likely visited and

the expected entry time for each one of them. When a mobile unit calls, the cell that is paged in the first place is the most likely one in the UMH. This is usually enough to find the terminal, even if this one had already performed a location update when it entered a location area.

In cases where the users do not behave according to their profiles, a standard technique is then utilized. Moreover, the procedures associated to the construction and use of the tables is simple enough to be executed in a mobile terminal: usually less than 100 comparisons or 2 update operations in a table.

CHAPTER III

PROPOSED STRATEGY FOR MINIMIZING ENERGY CONSUMPTION

The previous chapter introduced some of the existing strategies for maximizing lifetime of Object Tracking Sensor Networks (OTSNs). Energy consumption in the network can be reduced by switching off as many components as possible and as long as possible. Ideally, only the sensors which are closer to the mobile object should be active, and only while the object moves near them. One way to achieve this is by predicting the object's future location.

This work presents a new strategy to take advantage of the mobile objects' profile information in order to improve the energy consumption of the network.

3.1 Problem formulation

We look at the Object Tracking in Sensor Networks (OTSN) problem. We assume that a number of sensor nodes are deployed in a given area, called *monitored region.* The boundaries of this region are known by the applications that retrieve the information of interest about the moving objects. The sensor nodes are static, and the moving objects can also be seen as actuators, since they exchange information with the sensor nodes in order to take actions.

The sensing subsystem consists of a group of sensors and an actuator that link the node to the physical world. The computing subsystem consists of a microprocessor or microcontroller. The communication unit consists of a short range radio for wireless communication. The power supply subsystem houses the battery and the ac-dc converter and powers the rest of the node [23].

It is generally accepted that the communication unit has the greatest impact in the energy consumption. However, some works in the literature focus on the optimization of the sensing and computation costs.

In this work, each so called "node" is a logical representation of a group of physical sensors that together decide the mobile object's properties. In fact, the sensing of the mobile object's location and movement is a collaborative task that involves several sensors (in order to do triangulation, for example).

We make the following assumptions:

- A clustering scheme exists to form the groups called *clusters*. In effect, the sensor nodes we refer to in this work are the cluster heads or cluster leaders. The clustering is needed for the cluster head to know which sensors are under its responsibility and to be able to wake them up or accept their data. The clustering scheme is a pre-condition for our proposal, but its specifics are out of the scope of this work.
- The moving objects are electronically tagged and are identifiable in a unique way. This is necessary in order to have an individual profile for each object. If this assumption did not hold, the nodes could not recognize individual objects and the proposed algorithms would work in the same way for all the objects. The profiles would refer to an 'aggregate' object and would not reflect the individual behavior of the objects, thus reducing the efficiency of the scheme.
- The nodes must be synchronized with the application. This is necessary if we want a minimum quality of service (QoS) for the end user. The application needs to know the state of the monitored object at every T seconds. A certain additional delay can be allowed for the object missing cases in which a recovery mechanism has to be started. If this assumption did not hold and the synchronization was not possible, no QoS could be provided and the network would be useless.

- The sensor nodes must have knowledge of the remaining nodes' geographical location. This can be achieved by periodical messages exchanged between them. This issue is out of the scope of our work but it is necessary for the localization. The proposed algorithm will assume that the nodes are able to determine which node is closer to a given location.
- There exists a local coordinate system in which all sensors' positions are relative to a local, in-network reference. This precludes the need for GPS in individual nodes, which would increase their cost.

All the operations that are performed by a sensor network consume energy, be it transmission, reception or computation of data. Although the computational power of a sensor is expected to grow to follow Moore's law, battery technology is only expected to improve 2-3% per year [25]. The only way to maximize the network's lifetime is thus to use energy-efficient mechanisms to perform the foresaid operations.

At a sensor, the following relationships hold regarding the energy consumed at different functions [25]:

$$E_{Tr} = 2E_R \tag{3.1}$$

$$E_R >> E_{exec} \tag{3.2}$$

Where

E_{Tr} = Energy consumed in transmitting a packet

E_R = Energy consumed in receiving a packet

E_{exec}= Energy consumed per instruction-execution

Thus, a possible strategy to optimize energy is to reduce the number of high-energy operations (transmissions) at the cost of increasing the number of low-energy ones (instruction executions).

Two requirements of the application that monitors the region are the following:

1. *Sampling duration*: the sensor nodes need to sample the objects' movement for a given time, called "sampling duration". This depends on the specifics of the monitored phenomena. During this time, the MCU and the sensing unit are

active, but the radio components can be turned off [24] since no communication to the sink is going to take place.

2. *Reporting frequency*: similarly, the sink should be kept informed of the objects' movements with certain regularity, the *reporting frequency*. Only the nodes which detect the object in their area should activate their radio components and report to the sink when the end user requests so. The remaining nodes could have the radio off.

Under the above presented requirements, the problem we deal with is the maximization of the lifetime of the entire network by developing power-efficient algorithms to activate/deactivate the various components of the sensor nodes.

Requirements of the application: A network with S nodes samples the O moving objects during X seconds and has to report the information to the sink every T seconds.

Problem definition: Given the requirements for the object tracking application, formulate an energy saving scheme that minimizes the overall energy consumption of the OTSN constrained to an acceptable missing rate [24].

The missing rate is defined as the ratio of the number of missed reports to the total number of reports the application is supposed to receive from the network [24].

If a history record exists for the moving objects, built after a learning time and this is used to create profiles, the goal of the application is to know whether the objects are within or without their predicted (in the profiles) locations at any moment, with a granularity which is determined by the reporting frequency.

3.2 Proposed solution

We propose a profile-based strategy for saving power in OTSNs. The originality of this approach is the fact that it is the first time that profiles' knowledge is used in this type of application. The profile may contain several types of information on the object, depending on the specifics of the application. It could be only the zone location, or also its speed.

The learning process runs locally, at the cluster head level. The idea behind this approach is to save communication and sensing cost at the nodes. Sensing cost is saved when the object moves according to its profile since the node need not sense the characteristics of the movement. Communication cost is saved since one node does not have to send wake up messages to the next expected nodes because these already know when they have to wake up to sense the object. Ideally, if the object follows its profile at all times, the sensing and communication costs would be greatly reduced.

We hope to obtain better performance than other existing approaches, like prediction-based techniques.

3.3 Motivation for the use of profiles in OTSNs

The main goal of the energy-saving approaches in OTSNs is minimizing the number of transmissions and the amount of sensing time at the expense of more processing operations in the nodes. The use of profile information stored in-network can be efficient in achieving this goal. Instead of reporting the object's location at each one of the application's requests, the report will be sent only when the object's behavior differs from its stored profile (irregularities). Instead of having the nodes active all the time, they will be sensing and tracking for the object only when their movement characteristics are different from those known in advance. The degree of power savings will depend on the predictability of the moving objects, but we hope that this strategy can improve the network's lifetime for most OTSNs with reasonable mobile objects' predictability values. Since the sensor network can learn the objects' behavior (the nodes are densely distributed and close to the monitored phenomenon), the energy savings can improve in the long run.

The benefits of the profile information are multiple:

1. If the network knows the mobile objects' behavior, it is possible to allocate resources (actuators) where they can help to transmit data to the sink or other actors. Delay and bandwidth can be improved, since overheads are reduced.

2. It is also possible to improve the multi-hop transmission of data directly to the sink if the profile information is distributed in the network. The intermediate nodes will avoid unnecessary transmissions. Only the information that has been incorrectly predicted in the profiles will be reported.
3. The network itself becomes an important tool to build and update the profiles' database.
4. Compared to the prediction-based schemes, energy consumption is optimized, because the number of transmissions is minimized. The predictions need not be transmitted between nodes, since the network has the history information already.

Table 3.1 shows a summary of the advantages offered by each method regarding energy consumption. The first method (naïve) is the basic approach and consists in having all the nodes send any new readings to the sink. It is a reactive and event-driven method, while the prediction and profile-based ones are proactive and history-based.

Table 3.1 Comparison between several strategies for reducing energy consumption

	Transmissions at sensors	**Computation**	**Sensing**
Naïve	All new readings from sensors are sent to the sink	N/A	Always
Prediction-based	Only when new readings ≠ prediction	Calculate predictions at cluster heads	Only after detection, and during the sampling duration
Profile-based	Only when new readings ≠ information in profile	Update profiles and learning computations at cluster heads	Only when profile predicts presence of an object

In case of an irregularity in the object's behavior, two effects will follow:

1. An alarm would be triggered at the application. The network acts in this way as a control tool.
2. The profile must be updated.

The tradeoffs involved in the profile-based approach are:

1. The efficiency of the profile-based schemes depends on the mobile objects' own predictability. When the mobile objects do not follow the usual behavior, the recovery mechanisms' costs may overcome the gains because of the additional recovery operations.
2. The processing time associated to queries and updates in the profiles' database may be excessive in order to route real-time data. This downside also exists in prediction-based approaches, however.
3. Each cluster having a profile for a given object will activate itself whenever the probability that the object is under its zone is higher than a given threshold. Determining this threshold is a research issue. This threshold could also be adaptive to the actual network conditions, and be dynamically adjusted. A tradeoff is involved in this decision: the more zones to wake up, the higher the communication cost, but QoS is better.

3.4 Proposed profile-based strategies in OTSNs

We propose two profile-based approaches to track objects in an object-tracking sensor network. The first approach uses a global profile while the second approach uses a local profile.

3.4.1 Global profile based algorithm (GPBA)

The basic idea in this strategy is to replace continuous, real-time prediction by historical information (profile) on the mobile object's movement. The reports to the sink are skipped whenever the object stays within its profile. The sampling periods are not necessary if the object is detected inside its profile and the network knows its movement's characteristics.

In order for a node to wake up the nodes that will track the object in the upcoming cycle, it has to store the object's profile. This one must have been received from the object itself, which records the zones' ids and treats them in order to build its own profile. This profile information is transferred to the nodes it passes by during its

trajectory. This profile information can be built using statistical counting, neural networks, Bayesian networks or other machine learning techniques.

3.4.2 Local profile based algorithm (LPBA)

The second approach relies on the individual nodes to build their own local profile, by computing the probability that each mobile object be under their respective zone at any moment. Each node will thus be able to take the decision whether to wake up or not, regardless of the state of other nodes. Coordination exists in the previously detecting node in order to assure the QoS (periodic reports). This (the last host node id) is the only information needed to be stored by the mobile object. This approach cuts inter-nodes wake up transmissions costs. The sensing costs are however high, since the nodes stay awake for their entire profile duration even if the object has already been detected by another node. LPBA requires less intelligence built in the mobile object.

When the current node misses the object's position at a given cycle (by not receiving an acknowledge message before the expiration of a timeout) it starts the recovery mechanism.

Table 3.2 summarizes the differences between the approaches above presented.

Table 3.2 Global vs. Local approaches for profile-based OTSNs

	Global approach	Local approach
Communication costs between CHs	High (wake up messages from the TL)	Low (only the ACK messages)
Sensing costs	Low (only one node active after detection)	High (active nodes regardless of their probability to host the object)
Information in the object	Its global profile. A given node needs to receive its profile and those of its neighbors.	The last visited zone (that acts as target leader).
Acknowledge	Upon detection, the new TN sends a message back to the old TN based on its wake up message.	Upon detection, the new TN sends a message back to the old TN stored in the mobile object's tag.
QoS	Some missing reports are possible since not all the nodes in the profile will be active.	The missing report rate is minimal, since all the nodes in the profile will be trying to locate the object.
Synchronization	Not necessary. The 'wake up' messages do this task	Necessary periodically and among neighbor nodes.

3.4.3 Building the global profile

We explain here how the global profile used in GPBA is built. For every node (zone), we propose to divide the day in a given number of time slots of equal lengths. Each object has its own profile table. We will have as many profile tables as mobile objects are monitored by the network (assuming the objects are uniquely identifiable).

- When a node detects the presence of a mobile object, it updates its profile. The node keeps a count of the number of times that the object has been detected in each of the time slots (Table 3.3).
- The mobile object captures the profile table and counts the number of times that it stays under each zone for every time slot. After a certain learning period, an average probability of hosting the object can be deducted for every slot.

In Table 3.3, we see the data for the object's local profile in the area under a node *k* build the profile for the node. Each time slot will have an associated probability to host the object.

Table 3.3 Construction of a profile in one node

Node k						
		Slot 1	**Slot 2**	**Slot 3**	**Slot 4**	**Total for day**
Day 1	**Frequency**	3	4	7	1	15
	Probability	3/15	4/15	7/15	1/15	100
Day 2	**Cumulative Frequency**	3 + 1	4 + 5	7 + 8	1 + 2	15 + 16 = 31
	Probability	4/31	9/31	15/31	3/31	100
	Probability	P_{k1}	P_{k2}	P_{k3}	P_{k4}	

We can formulate the probabilities in Table 3.3 in terms of conditional probabilities:

P_{k1} = Pr (the object is detected in time slot 1 given that it is under node k) (3.3)

We need the inverse probability of the above, i.e., P_{1k}, the probability that the object will stay in each of the zones k for a given time slot, number 1 for instance. This would be useful for an algorithm that tries to predict the future location of the object. According to the Bayes' theorem, both probabilities are related:

$$P_{1k} = P_{k1} * \frac{\text{Pr (object being detected in slot 1)}}{\text{Pr (object being detected under zone k)}} \quad (3.4)$$

One of our assumptions is that the object has to be inside the monitored region at all times, thus the probability that it is inside any zone in one particular time slot is the same for all the slots, this is 1/(number of slots in which the day is divided).

The probability that, at any given time, the object stays under a particular zone depends on the mobility model. The most simplistic model, the random movement, would mean that all the zones have equal probability to host the object a priori. This probability would then be 1/(number of zones in which the monitored region is divided). An example of the resulting profile for an object O is given in Table 3.4.

Table 3.4 Profile at a moving object

Object O		
Time slot	**CH**	**P**
1	CH_1	P_{11}
	CH_3	P_{13}
2	CH_2	P_{21}
3	CH_5	P_{35}
	CH_3	P_{33}

In Table 3.4, CH = Id for a cluster head zone and P_{ij} = Probability that the object stays within the zone in CH_j in the time slot Ti.

The following relations hold for the profiles:

- The probabilities for a given time interval are sorted in a descending order and the minimum value is given by a threshold, which is a design parameter. This means that lower probabilities are not computed in the model

$$P_{ij} > P_{ik} > P_{th}, \text{ for all } i = 1 \text{ to } n, j,k < N \quad (3.5)$$

- The sum of the probabilities for a given zone must be at least 50%, in order for the profile information to be useful.

$$\sum_{j=1}^{ni} Pij > 0.5 \qquad (3.6)$$

The mobile object receives the local information of every node it encounters on its way and computes the global profile. This global profile is then transmitted back to the nodes, in case they needed the global view of the object's behavior to select the set of nodes that have to be woken up.

There is a tradeoff between the number of zones activated during the tracking stage and the communication cost to send them a wake up message upon the last detection. We call n the number of activated zones in the profile and therefore the number of communication messages sent.

3.4.4 Modeling the proposed approach

We consider several scenarios depending on whether the network can detect the mobile object based on the stored profile. In scenario 1, the mobile object is within its profile when it is detected by a sensor node. This means that the sink will not need to receive the report on the object's location at the end of the cycle. In scenario 2, the mobile object is out of its profile when it is first detected by a sensor node. This calls for a report to the sink at the end of the cycle. In both scenarios 1 and 2 the mobile object is tracked successfully at the end of the operating cycle (T seconds). No recovery mechanism is executed. Scenario 3 deals with the object missing situation; the object is not tracked by any of the sensor nodes at the end of the cycle and a recovery mechanism is started.

In the following sections we will introduce the two main strategies that we propose. The first one is based in the foresaid "global" approach and is called Global Profile Based Algorithm (GPBA). The second one is based on the "local" strategy and its name is Local Profile Based Algorithm (LPBA). We present the steps involved in both strategies.

3.5 Description of the Global Profile Based Algorithm (GPBA)

Scenario 1

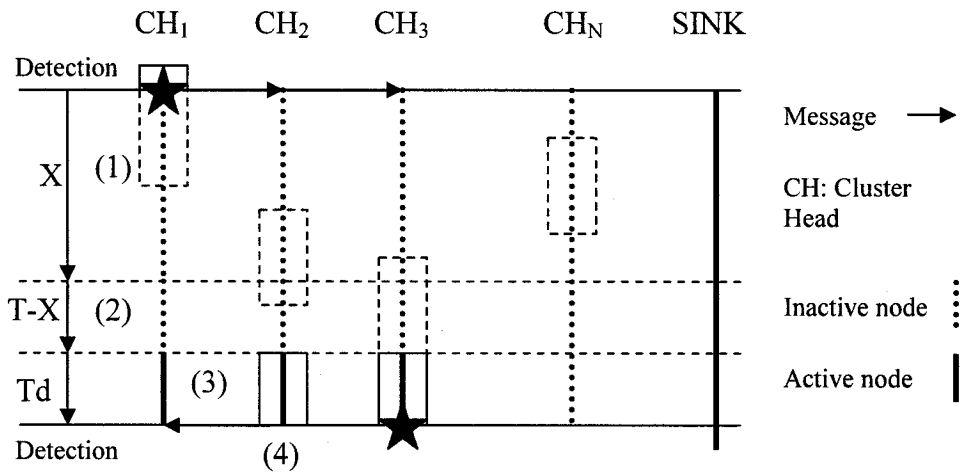


Figure 3.8 GPBA, scenario 1

The mobile object is within its profile at the first detection, so no report is sent to the sink T seconds later and it is not necessary to sample the object's movement. The current node will wake up T seconds later, along with a set of predicted nodes, to start sensing for the object in the next cycle.

The steps executed by the algorithm for this scenario shown in Figure 3.8, are:

1. The node that detects the object becomes the tracking leader (TL) and obtains the information on the profile of the object for its own zone and the neighboring zones which are predicted to detect the object in the following time interval.
2. Based on its local profile, the TL determines the optimum set of zones to be waked up T seconds later. The number of nodes that will be woken up is *n*. In order to calculate this number, the logic will balance the communication cost to wake them up with the activation (sensing) costs.
3. The selected nodes are waked up by the current TL.
4. All the nodes are sleeping for T seconds from the detection. No message is sent to the sink because the mobile object is within the profile.

5. After T seconds from the detection and (T-X) seconds sleeping, the predicted zones will wake up by themselves, along with the current node, and start tracking for the object. Their sensing units are active until the object is detected. At that point, the old target leader will go to sleep. The rest of the tracking nodes will continue sensing until the expiration of the timeout.
6. When the object is detected by one of the zones, an acknowledge message is sent back to the TL. This prevents the TL from starting a recovery process. The remaining active target nodes stay active and will eventually go to sleep state by themselves after the timeout T_{out}. This is to avoid extra communication costs which would cause 'turn off' messages from the TL. If the TL does not receive any acknowledge message before the expiration of the timeout, a recovery mechanism starts.

Scenario 2

The steps executed by the algorithm for this scenario, shown in Figure 3.9, are the same as in scenario 1, with the following two differences:

1. The current node samples the object's movement during X seconds.
2. The profile is updated at the current node and transferred to the mobile object. A message is sent to the sink because the mobile object is out of the profile.

Scenario 3

This scenario deals with the object missing situation. If after the cycle and a given extra time the object is not tracked yet, a recovery mechanism has to ensure the detection in the shortest possible time.

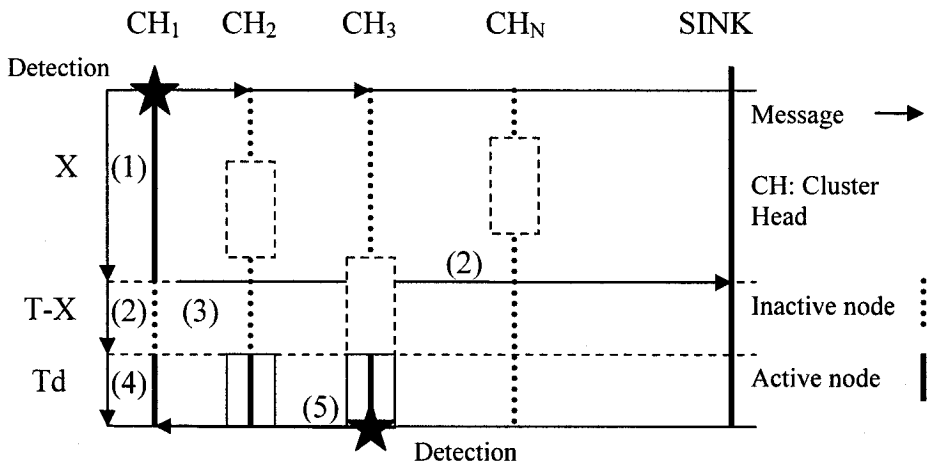


Figure 3.9 GPBA, scenario 2

In the PES algorithm, the zones neighboring the route from the current node to the predicted destination were woken up.This helped cover for the prediction errors in speed and direction. In the case of the profile-based algorithms, the prediction is multiple: a set of zones, possibly far away from each other, are woken up and try to track the object. We propose for this case to wake up the zones neighboring the target nodes for the recovery, due to the nature of the profile-based prediction, based on the behavior of the object (geographic information) rather than on the current speed or position (instant information). As usual, the TL waits for an acknowledge message from one of the nodes before the expiration of a timeout. The nodes stay awake for all this duration, except the one that detected the object. If the TL does not receive any ACK message, the second stage (a flooding recovery) is executed. It is sure that this stage will succeed in tracking the object, assuming it is always inside the monitored region.

3.6 Local Profile Based Algorithm (LPBA)

Scenario 1

One important issue in this approach is the determination of the time during which the nodes are active. Each node will autonomously decide when to activate its sensing

unit, based on its local probability. The steps executed by the algorithm for this scenario, shown in Figure 3.10, are the following:

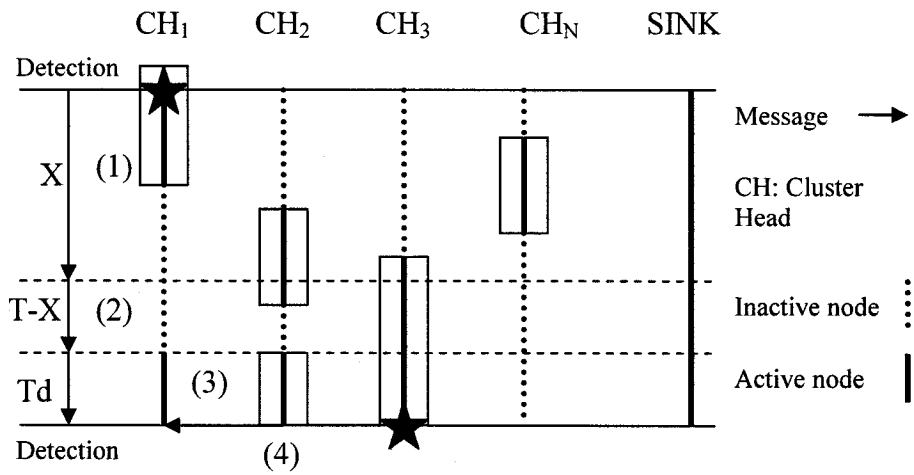


Figure 3.10 LPBA, scenario 1

1. The object is detected by one of the nodes in the profile (TL) that are active and sensing. The remaining nodes in the profile wake up by themselves to track the object if they decide so. In that case, they stay awake for a time that depends on the object's profile. The tracked mobile object records the id of the TL's zone as last visited zone.
2. On T, no message is sent to the sink because the mobile object is within the profile.
3. After T seconds from the detection and before the expiration of a timeout, one of the active zones in the profile will detect the object again. The mobile object transmits the identity of the old TL to the tracking node that has just detected it. The new TL reports back to the old TL in order to prevent it from starting a recovery process.

Scenario 2

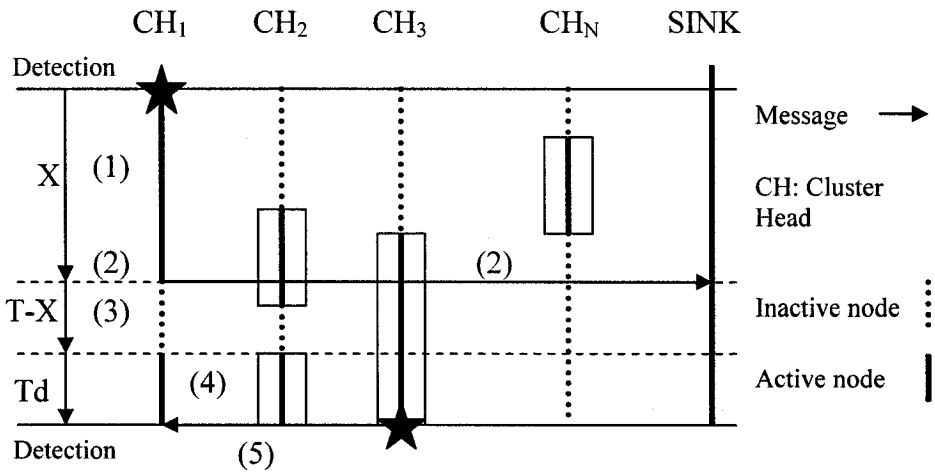


Figure 3.11 LPBA, scenario 2

The fact that the mobile object is out of its scenario means that sampling for X seconds is necessary and a message will be sent to the sink. The steps executed by the algorithm for this scenario, shown in Figure 3.11, are:

1. The object is detected by one of the nodes out of the profile (TL). The nodes in the profile have woken up by themselves to track the object. They stay awake for a time that depends on the object's profile. Besides, the TL stays awake for X seconds to sample the object.
2. On T, a report is sent to the SINK because the mobile object is out of the profile.
3. After T seconds from the detection and before the expiration of a timeout, one of the active zones in the profile will detect the object again. The mobile object transmits the identity of the old TL to the tracking node that has just detected it. The new TL reports back to the old TL in order to prevent it from starting a recovery process.

Scenario 3

The recovery mechanism is the same for both profile-based algorithms.

3.7 Analytical model

Our performance analysis uses power consumption as metric. We will compare the proposed profile-based scheme GPBA to the prediction-based scheme. We will also measure the number of object missing events (QoS).

Power consumption in a sensor node can be divided in three components [4]:

1. Sensing (Ps);
2. Communication (Pc);
3. Data processing (Pp)

We then model the power consumption in the following way:

$$P = Pc + Ps + Pp \tag{3.7}$$

The power consumed in data communication is generally considered to be the biggest of the three. Transmission and reception are nearly the same and the start-up power consumption in the transceiver circuitry should not be neglected [4]. When turning ON the transceiver, a large amount of power is consumed. This power consumption will be taken into account in the simulations.

We will not take into account the data processing component in the simulations. We will consider that the power required to perform the prediction calculations is the same as the one required for maintaining the profiles.

We will use the energy consumption model of WINS sensor nodes [10] as data for our simulation. The parameters are indicated in Table 3.5.

Table 3.5 Energy consumption on WINS nodes

Component	Mode	Energy Consumption (mW)
MCU	Active	360
MCU	Sleep	0.9
Sensor	Active	23
Radio	Transmission	720
Radio	Reception	369

In our work, an active node has both MCU and sensing units in the active state, so the energy consumption in that state is the combination of both data. More recent sensors allow the individual components of the nodes to be turned on separately.

We present in this chapter the results of preliminary tests carried out in MATLAB. In these tests we compare the performances of GPBA and LPBA to that of PES for a hypothetical operating cycle. It is an analytical study within the duration T seconds between two reports to the sink, plus the time extensions due to object missing events. In the simulations we consider one mobile object to be tracked and a number of logical sensor nodes (cluster heads).

The application running in the user end demands a report on the abnormalities in the objects' movements at every T seconds. The report must be sent to a special node that acts as gateway to external networks, called sink. The application considers an object missing after *Tout* seconds without having detected the object.

The following figure (Figure 3.12) depicts the network model we use to perform the performance comparisons between the different energy saving schemes across several scenarios. Some of the parameters explained below are displayed graphically in the figure.

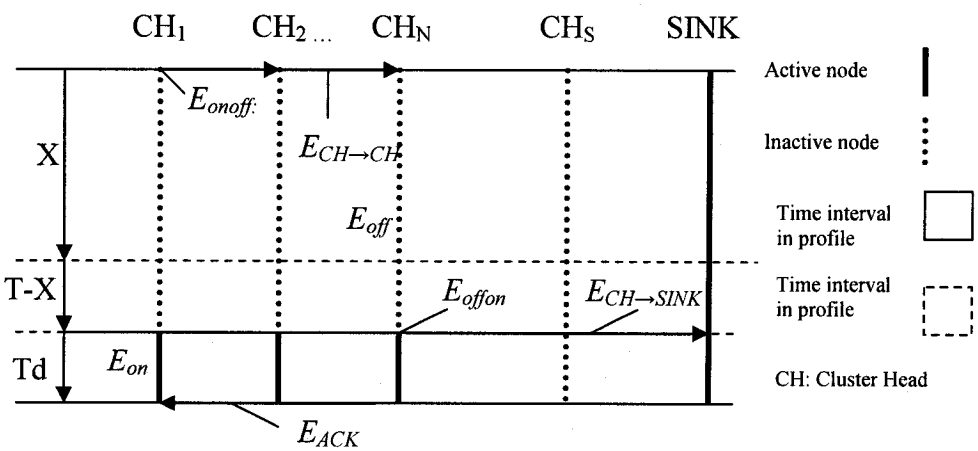


Figure 3.12 Parameters and the network model

The topology and the application parameters are modeled with the variables shown in Table 3.6.

Table 3.6 Topology and application parameters

S	Number of cluster heads in the network
T	Application period for reports to the sink
X	Sensing time in seconds
T_d	Time elapsed from the start of the tracking process until the object is detected
T_{out}	Maximum tracking time after which we consider a missing object. $T_d \leq T_{out}$
T_r	Time of first level of recovery, until object detection
T_f	Time of second level of recovery (flooding), until object detection

The energy consumption model reflects the fact that the sensor nodes consume energy in each of their functions and also in the mode transitions. We also take into account the energy consumed in performing the calculations to predict the future movement of the object, be it a prediction or a profile update (assumed to be equal in our simulations). These parameters are shown in Table 3.7.

Table 3.7 Energy consumption parameters

E_{on}	Energy consumption *rate* (per second) at a sensor node by having both computation and sensor components stay in active mode.
E_{off}	Energy consumption *rate* (per second) at a sensor node by having both computation and sensor components stay in sleeping mode.
E_{onoff}	Energy consumption in the switching from the awake mode to the sleep mode
E_{offon}	Energy consumption in the switching from the sleep mode to the awake mode
E_{pred}	Energy consumed to perform a prediction calculation at a node.
E_{update}	Energy consumed to perform a profile update at a node
$E_{CH \rightarrow CH}$	Energy consumption to send a message from the cluster head to each node of the set of target nodes to be waked up.
$E_{CH \rightarrow SINK}$	Energy consumption to send a message from a cluster head to the sink.

Since the multi-hop paradigm is considered for inter-nodes messages, the communication costs are the combination of the reception and transmission costs at the nodes on the route from the origin to the destination.

$$\{E_{CH\rightarrow CH}, E_{CH\rightarrow SINK}\} = E_{TX} + E_{RX} + \sum_{i \in Route} E_{TXi} + E_{RXi} \quad (3.8)$$

For both communication cost parameters, the number of nodes on the route will be set as a constant for the comparison between the different energy-saving schemes.
The current node sends a message to a target node because it believes this one will host the mobile object (T-X) seconds later. Therefore, $E_{CH\rightarrow CH}$ will be a function of the distance between the current target leader and each one of the target nodes.

The number of nodes on that route depends on the mobile object's speed and the radio coverage area for each node. We assume that the nodes have equal radio ranges and they are uniformly distributed along the route between the current node and the destination node. The number of nodes on the route will be then proportional to the object's speed and time interval until reaching the final position, and inversely proportional to the size of the monitored area associated to the node.
We can therefore write, as an approximation:

$$E_{CH\rightarrow CH} = (E_{TX} + E_{RX}) \cdot f(v \cdot (T\text{-}X)/R) \quad (3.9)$$

where:

v = speed of the mobile object [m/s]

T = reporting period [s]

X = sampling duration [s]

R = radio coverage per sensor [m]

Regarding the value of $E_{CH\rightarrow SINK}$, we can anticipate it will be larger than $E_{CH\rightarrow CH}$.. It will depend on the distance between the current node and the sink, which will be likely to be larger than the distance between two nodes inside the monitored region.

In order to characterize the profile and prediction's accuracies, we consider constant values for a cycle. We call profile accuracy the cumulative probability held by the set of nodes whose information is stored in the global profile. Table 3.8 outlines these parameters.

Table 3.8 Prediction and profile parameters

α	Accuracy of prediction: probability that the prediction is correct, for PES
γ	Cumulative probability that the mobile object is within its profile for a given operating cycle
n, z	Number of activated zones in the profile at an operating cycle (average) in the GPBA, LPBA schemes

In the MATLAB mathematical analysis, the probability for scenario 1 is the probability that the mobile object is within its profile, given that it has already been detected by a node. Its value is then the ratio between the number of zones in the profile and the total number of zones, k/S. Probability of scenario 2 is (1-k/S).

The probability of scenario 3 in the profile-based approach is the probability that the mobile object is not in any of the activated zones. It is thus the complement of the cumulative probability held by the profile, (1-γ), or, in case of the prediction-based approach, the complement of the prediction accuracy (1- α).

With regards to the recovery process, we need to model the probability that it will succeed in the first stage and the number of nodes that will be subsequently woken up. Table 3.9 describes the recovery mechanism's parameters.

Table 3.9 Recovery mechanism parameters

Par.	**Description**
β	probability that the first level of recovery succeeds to track the object
TN	Number of target nodes woken up by PES algorithm
RN	Number of nodes on the route from the current node to the predicted destination in PES
NN	Number of nodes neighboring the route from the current node to the predicted node in the PES algorithm
N	Number of nodes woken up as neighbors in the PBA algorithm

3.7.1 Analytical model for PES algorithm

In PES (see chapter II) only two scenarios exist:

1. Scenario 1: the mobile object is detected within the Tout interval.
2. Scenario 2: the mobile object is missed at Tout and a recovery mechanism is started. This second step necessarily tracks the object.

We calculate the average energy consumption in the operating cycle by adding up the different components of the energy in the different steps of the algorithm.

PES, scenario 1

1. During the sampling period (X seconds), the tracking node is sensing and active, while the rest of the nodes in the network are in sleep mode. The energy consumption is $E_{on} \times X + (S-1) \times E_{off} \times X$
2. After the sampling period, the current node has enough information to perform prediction as to which nodes will be woken up to track the object (the number of nodes to wake up is TN). It then sends wake up messages, as well as the report to the sink. The energy consumption is $E_{pred} + E_{CH \rightarrow SINK} + TN \times E_{CH \rightarrow CH}$
3. The current node turns off its sensing unit and all nodes are in sleep mode for (T-X) seconds. The energy consumption is $E_{onoff} + E_{off} \times S \times (T-X)$
4. The target nodes wake up and are sensing until the object is tracked after T_d seconds, while the remaining nodes in the network are in sleep mode. The energy consumption is $TN \times E_{offon} + TN \times E_{on} \times T_d + (S-TN) \times E_{off} \times T_d$; $T_d \leq T_{out}$
5. If the object has been tracked by one the target nodes (meaning the prediction was accurate, for which probability is α), the new current node sends an acknowledge message back to the previous tracking node. The energy consumption is $\alpha \times E_{TN \rightarrow CH}$

Therefore, the energy consumption expression in average for an operating cycle in scenario 1 is the following:

$$E_{PES,\,S1} = E_{on} \times \{X + TN \times T_d\} + E_{off} \times \{(S-1) \times X + S \times (T-X) + (S-TN) \times T_d\} + \\ + E_{pred} + E_{CH \rightarrow SINK} + E_{CH \rightarrow CH} + E_{onoff} + TN \times E_{offon} + \alpha \times E_{TN \rightarrow CH} \quad (3.10)$$

In the simulations with MATLAB, we consider $T_d = T_{out}$ for all the compared schemes. This is the worst case situation for all the schemes.

The value of TN depends on the heuristic chosen for the wake up mechanism. Table 3.10 shows the values used of the main parameters of the heuristic.

Table 3.10 Parameter values for a chosen heuristic

Heuristic	**TN: Number of target nodes to wake up**
DESTINATION	TN = 1
ROUTE	The value depends on the distance between the current node and the predicted node. TN = (RN+1)
ALL_NBR	The number of nodes neighboring the route can be approximated by considering that each node on the route has two neighbors (left and right sides), so TN = (3*RN+1)

PES, scenario 2

The recovery mechanism consists of two stages or levels. If the first level fails in tracking the object, the second level is executed, and it is sure that the mobile object will be tracked after this stage. In the first level of recovery, the heuristic ALL_NBR is used if it has not been already the case in scenario 1. Otherwise, the same nodes stay awake. The number of neighboring nodes that are active in the first stage of the recovery mechanism is noted NN, and its possible values are listed on Table 3.11. If the heuristic DESTINATION is used in the first stage of the recovery, both route and neighboring nodes will be woken up in the second stage.

Table 3.11 Number of neighbor nodes to wake up in first stage of recovery (PES)

Heuristic used for the wake-up	**NN: Number of target nodes that are active in the first level of recovery (including previously active nodes)**
DESTINATION	NN = (3*RN+1)
ROUTE	NN = (3*RN+1)
ALL_NBR	NN = TN

1. First level of recovery:

   The current tracking node sends wake up messages to the neighbor nodes that are not yet active. These ones turn their sensing units on and start trying to track the object. If one succeeds (probability β), it sends back an acknowledge message to the previous current node. The energy consumption is:

$$E_{PES,\ recovery1} = (NN\text{-}TN) \times E_{CH \rightarrow CH} + (NN\text{-}TN) \times E_{offon} + (NN+1) \times E_{on} \times T_r + (S\text{-}NN\text{-}1) \times E_{off} \times T_r + \beta \times E_{RN \rightarrow CH} \quad (3.11)$$

2. In the second stage of the recovery, all the nodes in the network that were not yet active turn on their sensing units, tracking the object for T_f seconds until the object is tracked (sure event since we assumed that the object is always inside the monitored region, which is covered completely by the nodes). The energy consumption is:

$$E_{PES,\ recovery2} = (S\text{-}NN\text{-}1) \times E_{CH \rightarrow CH} + (S\text{-}NN\text{-}1) \times E_{offon} + S \times E_{on} \times T_f \quad (3.12)$$

If the first level of recovery does not succeed in tracking the object, the elapsed time for the first stage will have attained a certain timeout. In the simulations we will consider it to be the same as in scenario 1 (T_{out}). Thus, the global average energy consumption for an operating cycle in PES is as follows:

$$E_{PES,\ S2} = \beta \times E_{recovery1} + (1-\beta) \times (E_{recovery1(Tr=Tout)} + E_{recovery2}) \quad (3.13)$$

The probability that the recovery mechanism is necessary is (1- α); thus, the average energy consumption for the PES algorithm is calculated as follows:

$$E_{PES} = E_{PES,\ S1} + (1-\alpha) \times E_{PES,\ S2} \quad (3.14)$$

3.7.2 Analytical model for GPBA algorithm

In GPBA, three scenarios exist:

1. Scenario 1: the mobile object moves according to its profile at the beginning of the operating cycle and its location is detected before Tout.
2. Scenario 2: the mobile object is outside its profile at the beginning of the operating cycle and its location is detected before Tout.
3. Scenario 3: the mobile object is not detected before Tout, at the end of the operating cycle, and a recovery mechanism is executed to track it.

We calculate the average energy consumption in the operating cycle by adding up the different components of the energy in the several steps of the algorithm, as shown in Figure 3.8.

In the following we show the details of the energy consumption calculations for scenarios 1 and 2 for our proposed algorithm GPBA. Figure 3.8 showed the steps involved for scenario 1.

GPBA, scenario 1

1. The current node receives the profile information from the mobile object and turns off its sensing unit (it follows its historical behavior). It sends wake up messages to the next predicted target nodes according to the profile. The energy consumption is: $E_{CH\rightarrow object} + E_{profile} + E_{onoff} + n\times E_{CH\rightarrow CH}$
2. The target nodes receiving the wake up messages turn their sensing units on. The energy consumption is: $n\times E_{offon}$
3. All nodes in the network have their sensing units deactivated during these T seconds. The energy consumption is: $S\times E_{off}\times T$
4. The current node and the target nodes wake up and track the object for T_d seconds. The energy consumption is: $(n+1)\times E_{offon} + (n+1)\times E_{on}\times T_d + (S-n-1)\times E_{off}\times T_d$
5. All the nodes go to sleep except the new current node. The new current node sends an acknowledge message back to the previous tracking node, and the remaining

nodes stay active until a timeout expires. The energy consumption is then: $E_{CH\rightarrow CH}$ + $n\times E_{onoff}$ + $(n-1)\times E_{on}\times (T_{out} - T_d)$

The overall energy consumption for this scenario is therefore:

$$E_{GPBA,\,S1} = E_{CH\rightarrow object} + E_{pred} + (1+n)\times E_{onoff} + (n+1)\times E_{CH\rightarrow CH} + (2n+1)\times E_{offon} + (S\times T + (S-n-1)\times T_d)\times E_{off} + ((n-1)\times (T_{out} - T_d) + (n+1)\times T_d)\times E_{on} \quad (3.15)$$

In scenario 2, the object is out of its profile at the beginning of the cycle, which means that the object must be sampled and a report must be sent to the sink. Figure 3.9 showed the steps involved.

GPBA, scenario 2

1. The new tracking node senses the object for X seconds and updates its profile. The energy consumption at this step is: $E_{on}\times X + (S-1)\times E_{off}\times X + E_{profile}$
2. The new updated profile is transferred to the object. The energy consumption is: $E_{CH\rightarrow object} + E_{update}$
3. All the nodes go then to sleep and the next target nodes are determined from the profile stored in the node; the report is sent to the sink. The energy consumption is: $S\times E_{off}\times (T-X) + n^* E_{CH\rightarrow CH} + E_{CH\rightarrow SINK}$
4. The chosen target nodes wake up and try to track the object until detection after T_d seconds. The energy consumption is: $(n+1)\times E_{offon} + (S-n-1)\times E_{off}\times T_d + (n+1)\times E_{on}\times T_d$
5. The new tracking node sends an acknowledge message back to the old tracking node and the non-detecting tracking nodes stay in active nodes until expiration of the timeout. The energy consumption is: $E_{CH\rightarrow CH} + n\times E_{onoff} + (n-1)\times E_{on}\times (T_{out} - T_d)$

The overall energy consumption for this scenario is therefore:

$$E_{GPBA,\,S2} = E_{on}\times(X + (n+1)\times Td + (n-1)\times (T_{out} - T_d)) + E_{off}\times ((S-1)\times X + S\times(T-X) + (S-n-1)\times Td) + E_{pred} + (n+1)^* E_{CH\rightarrow CH} + E_{CH\rightarrow object} + E_{update} + E_{CH\rightarrow SINK} + (n+1)\times E_{offon} + n\times E_{onoff} \quad (3.16)$$

GPBA, scenario 3

As shown in Figure 3.13, assuming a square-shaped radio coverage area for a node (only for clarity purposes), the set of awake nodes is different depending on the selected approach, either prediction-based or profile-based. Figure 3.13 a) shows the active nodes in the recovery mechanism for a profile-based algorithm while Figure 3.13 b) shows the active nodes for the PES strategy. The number of activated zones does not depend on the distance between current node and target node, or the speed and direction of the mobile object in the case of the profile-based approach, whereas it does depend on all those factors in the PES algorithm.

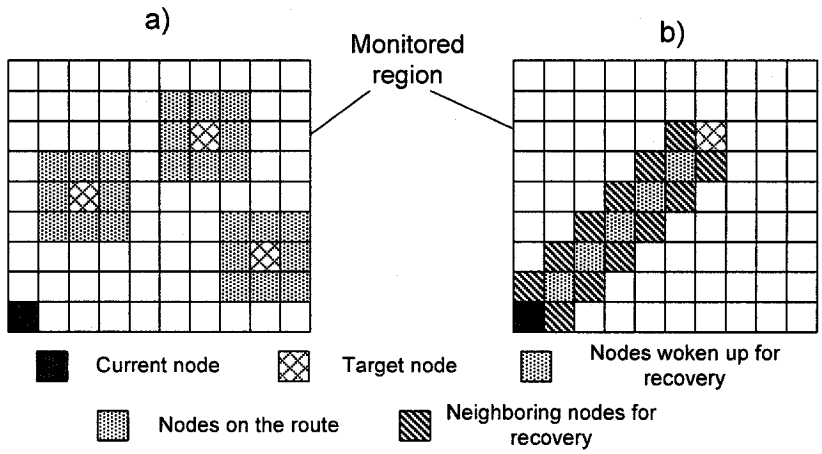


Figure 3.13 Active nodes for the first step of the recovery mechanism

Assuming the profile has k zones at the time of the object missing, the number of neighboring zones NN that will be woken up by the TL for the first stage of the recovery process can be calculated.

The two steps for the recovery mechanism incur in the following costs:

1. In the first stage of recovery, the nodes neighboring the nodes in the profile are woken up and stay active, trying to locate the object until expiration of a timeout. If they succeed, an acknowledge message is sent back to the previous tracking

node in order to prevent the second stage of recovery. The energy consumption is:

$$E_{GPBA,\ recovery1} = NN \times E_{CH \rightarrow CH} + NN \times E_{offon} + (NN+1) \times E_{on} \times T_r + (S\text{-}NN\text{-}1) \times E_{off} \times T_r + \beta \times E_{RN \rightarrow CH} \quad (3.17)$$

2. If the second stage of recovery is needed, all the remaining nodes in the network are woken up and stay active until a timeout T_f. The probability of tracking the object eventually at this stage is 100% assuming total covering of the monitored region. The energy consumption is:

$$E_{GPBA,recovery2} = (S\text{-}NN\text{-}1) \times (E_{CH \rightarrow CH} + E_{offon}) + S \times E_{on} \times T_f \quad (3.18)$$

Therefore, the global expression for the energy consumed in the recovery process for the GPBA algorithm is the following:

$$E_{GPBA,recovery} = E_{GPBA,recovery1} + (1-\beta) \times E_{GPBA,recovery2} =$$
$$= (NN + (1-\beta) \times (S\text{-}NN) + \beta) \times (E_{CH \rightarrow CH} + E_{offon}) +$$
$$+ (NN \times T_r + S \times T_r \times (1-\beta) \times (S\text{-}NN)) \times E_{on} + (S\text{-}NN) \times E_{off} \times T_r \quad (3.19)$$

In both scenarios 1 and 2, the mobile object is tracked at the end of the cycle before the timeout expiration. μ is the probability that the object is under one of the n zones which were woken up by the TL after computations. Scenarios 1 and 2 will happen no matter if the recovery is necessary or not.

$$E_{GPBA} = (k/S) \times (\gamma \times E_{GPBA,\ S1} + (1-\gamma) \times E_{GPBA,\ S2}) + (1-k/S) \times (\gamma \times E_{GPBA,\ S1} + (1-\gamma) \times E_{GPBA,\ S2}) + (1-\mu) \times E_{GPBA,recovery} \quad (3.20)$$

$$E_{GPBA} = k/S \times E_{GPBA,\ S1} + (1-k/S) \times E_{GPBA,\ S2} + (1-\mu) \times E_{GPBA,recovery} \quad (3.21)$$

3.8 Analytical results

In this section we show the results of the tests that compare the analytical performance of the GPBA to PES. Interested readers can find the results for LPBA in appendix. We don't present results for comparisons with the basic method, naïve (which keeps all nodes active all the time), because PES is already better than naïve. The tests have been conducted in MATLAB and are based on the analytical models that were introduced in section 3.7. In Table 3.12 we describe the parameters that have been used in the tests.

Table 3.12 MATLAB analysis parameters

Parameter	Value	Parameter	Value
S	95 nodes	E_{pred}	30 mW
T	1 second	E_{update}	30 mW
X	100 ms	$E_{CH \rightarrow CH}$	1415.7 mW
T_d	T_{out}	$E_{CH \rightarrow SINK}$	9800 mW
T_r	T_{out}	α	(0.5, 1)
T_f	T_{out}	γ	(0.5, 1)
T_{out}	300 ms	k	varies
T_p	0.5	n	varies
E_{on}	383 mW	μ	*varies*
E_{off}	0.9 mW	β	0.6
E_{onoff}	25 mW	*Mmonitored region*	120 m radius
E_{offon}	50 mW	v	5 m/s
		R	15 m

Topology

We suppose a network of S=95 logical sensor nodes evenly placed in a circular monitored area of radius 120 m. The sensor nodes have hexagon-shaped detection areas with six exact neighbor nodes each. The radio range of each node is 15 m.

Application requirements

We assume that the sensor nodes need to sample the environment for X=100 ms and that the application has to keep up to date the location information about the object

every second (T=1). The acceptable timeout to track an object is 300 ms (this will be the timeout period after which the recovery mechanism will start).

Energy consumption parameters

In our work, an active node has both MCU and sensing units in the active state, so the energy consumption in that state is the combination of both data. Hence,

$$E_{on} = 360 + 23 = 383 \text{ mW, and } E_{off} = 0.9 \text{ mW} \quad (3.22)$$

As explained before in this document, $E_{CH \rightarrow CH}$ = (720+369)*(r+1), where r = number of nodes on the route between the current node and the target node.

$$r \approx (v \cdot (T - X) / R) = (5 \text{ m/s} \cdot 0.9 \text{ s})/15 \text{ m} = 0.3 \quad (3.23)$$

$$(r + 1) \approx 1.3 \quad (3.24)$$

$$E_{CH \rightarrow CH} \approx (720+369) * 1.3 = 1415.7 \text{ mW} \quad (3.25)$$

The real value will be larger than that, since the multi-hop routing will force more than one node to relay the packets between origin and destination, for each pair of intermediate nodes on the route. $E_{CH \rightarrow SINK}$ follows the same function, but in this case the number of nodes on the route depends solely on the distance from the node to the sink. As an average value we will consider that the node is the center of the monitored region and that the sink is on the border. Using the same proportionality constant as for $E_{CH \rightarrow CH}$, we obtain:

$$E_{CH \rightarrow SINK} \approx (720 + 369) * (1 + r), \text{ where } r \approx 120/15 = 8 \quad (3.26)$$

Therefore,

$$E_{CH \rightarrow SINK} \approx 9800 \text{ mW} \quad (3.27)$$

We assume a value for the switching mode power consumption off-on of 50 mW, and half that value for the transition from on to off. The energy consumption per instruction execution will be modeled as being 10 times smaller than that for reception, so E_{pred} and E_{update} = 30 mW. We are assuming that the computation cost to update the profile, when necessary, is the same as to perform prediction calculations (a pessimistic scenario).

Prediction and profile accuracies

To compare the approaches in a fair way, we will consider that the number of zones waked up by the profile-based algorithm is the same as that waked up by the prediction-based one. And the accuracy of the prediction will be the same as the combined probability of the activated zones in the profile-based scheme.

In the following, we present the results for the tests of the analytical model which have been performed in MATLAB. Interested readers can look at Appendix A for details on test results for scenarios 1, 2 and 3 alone. We present some summarized results for those tests here, as well as the results for all the scenarios considered together in detail

Performance comparison for scenario 1

Table 3.13 outlines the average performance gains of GPBA, for a variable ratio of nodes on the PES route to active nodes in the profile between 50% and 150%. The table shows that the improvement can reach 60% if fewer than 5 nodes are needed in the profile for the next cycle and there are more nodes on the route to the target than there are in the profile. Even with more nodes in the profile than on the route, GPBA achieves an improvement of nearly 20% in average. The results don't change much if more sampling time X is needed.

Table 3.13 GPBA/PES relative performance for several (RN/n) ratios in scenario 1

Nodes in the profile, n	<5	[5,15]	>15
Gain, RN = 0.5 n	40%	20%	-
Gain, RN = n	60%	40%	20%
Gain, RN = 1.25 n	60%	40%	25%
Gain, RN = 1.5 n	60%	50%	40%

Performance comparison for scenario 2

When the mobile object is detected by a node out of its profile at the beginning of the cycle, the tracking node has to sample its movement, as in PES. Besides, a report message has to be sent to the sink.

Figure 3.14 shows the results of the analysis when the number of zones $RN = n$. It indicates that GPBA outperforms PES for 10 nodes in the profile or less, when the node-object communication cost $E_{node->object}$ is 20% or less than the inter-cluster head communication cost, $E_{CH->CH}$. The gain is small for this scenario 2: about 10% at best.

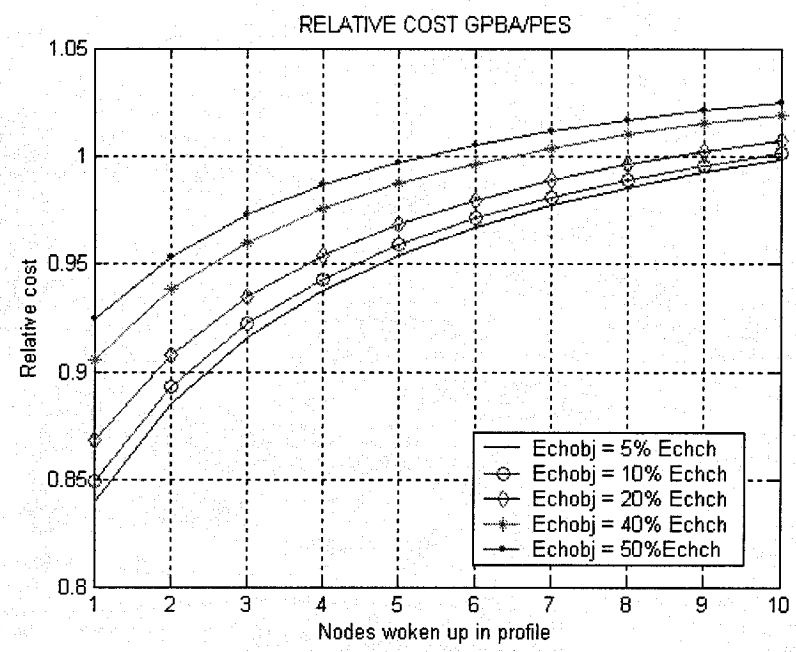


Figure 3.14 GPBA/PES for scenario 2

Also in Figure 3.14 we notice that improvements in node-object communication cost beyond 20% do not provide significant performance gains: the relative performance is almost equal for 5%, 10% and 20% (node-object / node-node) ratios.

Performance comparison for scenario 3

Table 3.14 outlines the several steps involved in the recovery mechanism for the algorithms studied. We can observe the factors impacting the different costs of the first stage of recovery for each one of the algorithms. The second stage is the same for both approaches (flooding recovery).

Table 3.14 Comparison of recovery mechanisms for the compared algorithms

1st stage recovery	PES	GPBA
Nodes that will be activated	Nodes neighboring the nodes on the route (*RN*)	Nodes neighboring the n active nodes of the profile
1st recovery stage cost magnitude	Nodes woken up: $2\times RN$ Total active nodes: $3\times RN$	Nodes woken up: $6\times n$ Total active nodes: $7\times n$

In PES, the cost for the first stage of the recovery mechanism depends on the distance from the current node to the predicted next nodes (or on the measured speed of the object), since this distance determines the number of nodes that will be woken up and stay active, tracking for the object; in GPBA it is the number of zones that belong to the profile what determines the number of nodes that will be woken up for the recovery, as shown in the table. We will run simulations for a series of values for n on the x-axis, and a set of specific values for RN. Interested readers can look at results and figures for the analytical tests in Appendix A.

Performance comparison

In this sub-section we perform tests with all the scenarios considered, combined according to their individual probabilities. Figure 3.15 shows the relative cost GPBA/PES for several possibilities of number of nodes. In this test, the number of active nodes is the same for both approaches; this means that the number of nodes on the route to the destination in PES is the same as the number of active zones belonging to

the profile in GPBA. GPBA is more cost-efficient than PES when the probability that the mobile object is within its profile is greater than 85% as long as there are less than 6 nodes in the profile. The more nodes in the profile, the greater this probability needs to be in order to outperform PES. For predictabilities over 95%, even 10 nodes in the profile can provide a performance gain relative to PES.

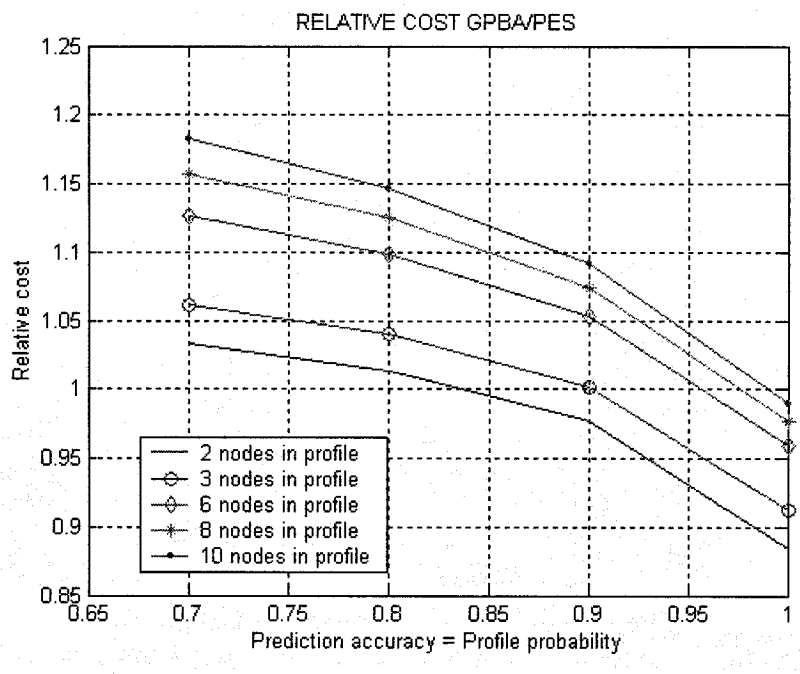


Figure 3.15 GPBA/PES for varying number of active nodes in profile

When there are fewer nodes in the active profile than there are on the route to the predicted destination, GPBA outperforms PES clearly. Figure 3.16 shows that pairs (RN = 3, n = 7) and (RN = 4, n = 6) yield an average energy consumption improvement of 20%, whereas for an equal number of active nodes GPBA needs 95% probability to host the mobile object in the profile to be more energy-efficient than PES. The probability of success in the first stage of recovery and the sampling duration are set equal in the simulations for both approaches.

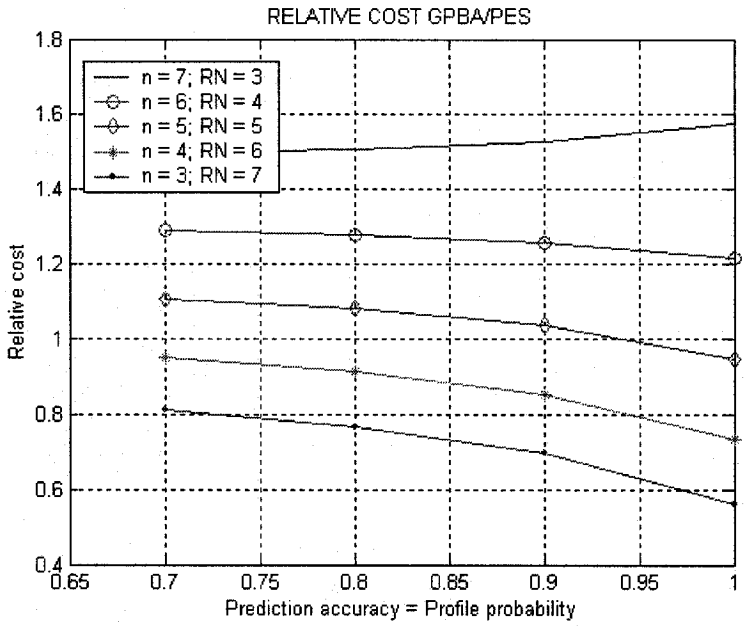


Figure 3.16 GPBA/PES for varying ratio of active nodes in GPBA, PES

CHAPTER IV

IMPLEMENTATION AND PERFORMANCE EVALUATION

In the previous chapter, we have proposed two power-saving algorithms to schedule activity in Object Tracking Sensor Networks. This chapter presents the implementation details and data structures of our proposed algorithms, besides the specifics of the implementation and simulation environment. Afterwards we will present our simulation results based on an experiments' plan. Finally we will analyze and compare our proposed algorithm's performance relative to other algorithms already existing in the literature.

4.1 Implementation and simulation environment

This section introduces the development environment that has been used to implement our solution. We will present here the main characteristics of the simulation software which we employed.

4.1.1 The development environment

We used a PC platform with an *Intel Pentium IV* processor 3 Gigahertz (GHz) and 1 GB RAM. The operating system is *Windows XP Professional*, from *Microsoft*, to which *service pack 2* has been added. The development and compilation of the source code were performed through the *Visual C++ 6.0* editor and the *nmake* engine, respectively, both belonging to the *Visual Studio .NET* (2003) suite form *Microsoft*.

4.1.2 QualNet: a discrete-event simulator

The QualNet environment is a simulator especially conceived for simulating ad hoc wireless networks in which there is a mobility component. It is the commercial

version of *GloMoSim* (*Global Mobile system Simulator*) developed by *University of California at Los Angeles* (UCLA). QualNet and GloMoSim are based on *PARSEC* (*PARallel Simulation Environment for Complex system*), a discrete-events parallel simulation language similar to C. Each one of the protocols is conceived as automata where every event modifies the protocol and the system's states.

The simulator has a layer-based architecture to allow for easier to implement extensions to the source code. The layers are: application, transport, network, link and physical. Discrete-time events are generated by the simulator and simulate the communication between same layer's protocols. It is also possible to communicate between layers directly, bypassing lower layers.

In order to add a new protocol to Qualnet, the source files have to be modified besides the addition of new ones. The executable file *qualnet.exe* is then compiled as a C-written program. The characteristics of the simulation can then be simulated. Results are collected on text files.

4.2 Implementation details

We have implemented two protocols, PES and our proposed algorithm GPBA. We will discuss in this section the implementation details of both, including the necessary data structures and the functions we have used.

4.2.1 Implementation strategy in *QualNet Simulator*

The file *node.h* has been modified to save statistical information that will be useful for the performance measure of the protocols: the energy consumption so far, the number of times the node has been tracking the object and has missed it, the number of transmissions and receptions and the running time its sensing and computation unit has stayed in active mode. Information saved in this file is also a Boolean variable used to identify the node as being the closest to the mobile object (and in charge of its tracking).

4.2.2 Data structure and functions

Data structures are defined in files *node.h* for the nodes' data that will be updated all along the program's execution, and *pes.h* or *gpba.h* for the implementation of both protocols. The figures in this section present those structures.

Figure 4.1 presents the statistical data associated to the nodes and useful for the performance metrics comparison. Besides, the variables *IamTL* and *IamClosest* hold the information on whether the node is currently playing the role of tracking node (target leader), and in this case is set to TRUE, or not, and whether the node is the closest one to the mobile object or not, and in case it is, it is supposed to be the only node that effectively detects it. This is to simulate the fact that only one cluster head (node) can detect the object at all times. The mobile object transmits with the same characteristics as the other nodes, and the radio range can reach more than one node.

Variables *BatteryEnergy*, *missings*, *stime*, *rxs*, *txs* and switches store, respectively, the running battery energy that has been consumed so far by the current node, and is updated at each time the node performs an operation, be it a transmission, a reception, mode switch (on to off, off to on). *Missings* will store the number of times the object has been missed at the application's reporting period and a location information report has failed to be sent; *stime* stores the time during which the node has been in active mode, while *rxs* and *tx* store, respectively, the number of receptions and transmissions. The variable *LastEventTime* will help determine the elapsed time between the current time and the last time an event happened, and it is used to calculate the energy consumption of time-dependent operations, such as the energy consumed for being in the active mode. *inProfile* is a variable that stores the information on whether the mobile object has just been detected is moving within its profile or not.

```
BOOL  IamTL;
BOOL  IamClosest;
// energy accountancy
Double BatteryEnergy;
int   missings;
clocktype   stime;
int   rxs;
int txs;
int switches;
BOOL  inProfile;
clocktype   LastEventTime;
```

Figure 4.1 Constants defining energy consumption

Figure 4.2 shows the energy consumption parameters we have used, based on the data from the WINS nodes. The operations that have been considered appear there. Some of these variables imply the measure of the time interval we have to consider to account for the consumed energy. This is the case of the active time energy consumption; others need only a counter of times that a transmission or a reception has happened.

```
#define BATTERY_TX_CONSUMPTION       720
#define BATTERY_RX_CONSUMPTION       369
#define BATTERY_IDLE_CONSUMPTION     1
#define BATTERY_ACTIVE_CONSUMPTION   383
#define BATTERY_ONOFF_CONSUMPTION    30
#define BATTERY_OFFON_CONSUMPTION    50
```

Figure 4.2 Constants defining energy consumption

Figure 4.3 shows the parameters for the network's topology and the distance between nodes. This distance is the minimum one between nodes. It is used to determine one node's neighbors in a geographical way, in order to designate the nodes that will be woken up for a recovery mechanism, for example.

```
#define NUM_OF_NODES 31
#define NEIGHBOR_DISTANCE 1000
```

Figure 4.3 Constants defining topology

Figure 4.4 shows the parameters for the characteristics of the profile's time limits. To build the profile, our protocol divides the day in slots, supposing that the time frame for the object's regularity extends itself during all the day. If we knew that the object's routine behavior lasts for a half day, we would have to divide that half day in equal intervals and then convert the remaining hours of the day to the same time scale. This is the goal of the last constant, `TIMESLOT_NORM_LIMIT`, as upper limit for the predictable behavior's time.

```
#define NUM_OF_INTERVALS 10
#define TIMESLOT_LIMIT_1 15
#define TIMESLOT_LIMIT_2 25
#define TIMESLOT_LIMIT_3 35
#define TIMESLOT_LIMIT_4 45
#define TIMESLOT_LIMIT_5 55
#define TIMESLOT_NORM_LIMIT 65
```

Figure 4.4 Constants defining profile's parameters

Figure 4.5 shows the parameters for the timers' lengths in seconds:

- `APPLICATION_T`: the reporting period, this is the time periodicity with which the sink is supposed to receive a report from the network with the mobile object's location information.
- `SAMPLING_DURATION`: the time to sample the object's movement, this is to capture its characteristics such as speed or location. The longer this time is for the same reporting period, the shorter the uncertainty time (T-X) during which the mobile object could flee and thus the less the probability to miss it when the nodes will wake up again to track the object.
- `PROCESSING_DURATION`: It is the length (T-X) during which the target leader is in the Sleep mode for its computation and sensing units, waiting to wake up again to track the object.

- `DETECTING_DURATION`: time length during which the target leader expects to receive an acknowledge message from one of the tracking nodes that have been woken up saying that the object has been detected, or also when it expects to detect itself the object. This is the "timeout" after whose expiration the recovery mechanism is started.
- `WAITING_DURATION`: the time length during which a tracking node to be stays in sleep mode, waiting to wake up. It is usually equal to `DETECTING_DURATION`.
- `ACTIVATION_DURATION: the waiting time for tracking nodes in the recovery mechanism.`
- `RECOVERY_DURATION`: the timeout for a tracking node taking part in a recovery phase. After this time, the node goes to sleep if it has not yet detected the object.
- `SECOND_RECOVERY_DURATION`: the timeout for a tracking node taking part in a second recovery phase. After this time, the node goes to sleep if it has not yet detected the object. With the underlying assumptions of our model, this timeout will not expire if the object remains inside the monitored area.
- `TRACKING_DURATION`: the timeout for a tracking node taking part in a normal cycle tracking operation. After this time, the node goes to sleep if it has not detected the object yet.
- `TL_RECOVERY_DURATION`: the timeout for the target leader to consider the object has been missed in the first recovery phase. If this is the case, the second recovery is executed.

```
// timers Secs
#define APPLICATION_T          10
#define SAMPLING_DURATION      8
#define PROCESSING_DURATION        2
#define DETECTING_DURATION     3
#define WAITING_DURATION       2
#define ACTIVATION_DURATION        5
#define RECOVERY_DURATION      10
#define SECOND_RECOVERY_DURATION 10
#define TRACKING_DURATION      3
#define TL_RECOVERY_DURATION       10
```

Figure 4.5 Constants defining the application's timers

Figure 4.6 is a data structure to store the last two positions of the mobile object. It is used for PES' prediction algorithm, that considers that the mobile object will follow the same direction and speed for the following (T-X) seconds and infers the future position from that information, by keeping the same tangent angle.

```
typedef struct {
   double x;
   double y;
} last_two;
```

Figure 4.6 Data structure containing the last two mobile object's positions

Figure 4.7 shows the parameters for the characteristics of the profile. The profile will store the three (at a maximum) highest-probability zones where the object is expected to be for each one of the time slots the day has been divided in. Each time slot consists of:

- The time slot limits start and end, `time1` and `time2`.
- The node IDs (QualNet Ids) that identify uniquely the node.
- The relative frequency with which the mobile object has been in the current node's detection area during the current time slot. This frequency is updated continuously and is the basis for the calculations of probability.

- The number of times the mobile object has been detected under each one of the nodes' detecting areas. Only two nodes are not included in this computation: the mobile node and the sink. The mobile object will compare these frequencies and choose the three highest ones to be included in the three top positions of the profile. It is a static criterion to choose the nodes that will be woken up.

```
typedef struct {
    int time1;
    int time2;
    int nodeId_1;
    int freq_1;
    int nodeId_2;
    int freq_2;
    int nodeId_3;
    int freq_3;
    int distribution[NUM_OF_NODES-2];
} profile;
```

Figure 4.7 Data structure containing the mobile object's profile

4.3 Implementation of the protocols

In this section we will present the details for the implementation of the main reference used to compare our algorithm, the PrEdiction-based Scheme [22], as well as the proposed protocol, GPBA.

The protocols are implemented on the application layer, using UDP as transport protocol and the QualNet API function *APP_UdpSendNewData* .

Figure 4.8 show the algorithm to be executed at every node to run the PES algorithm.

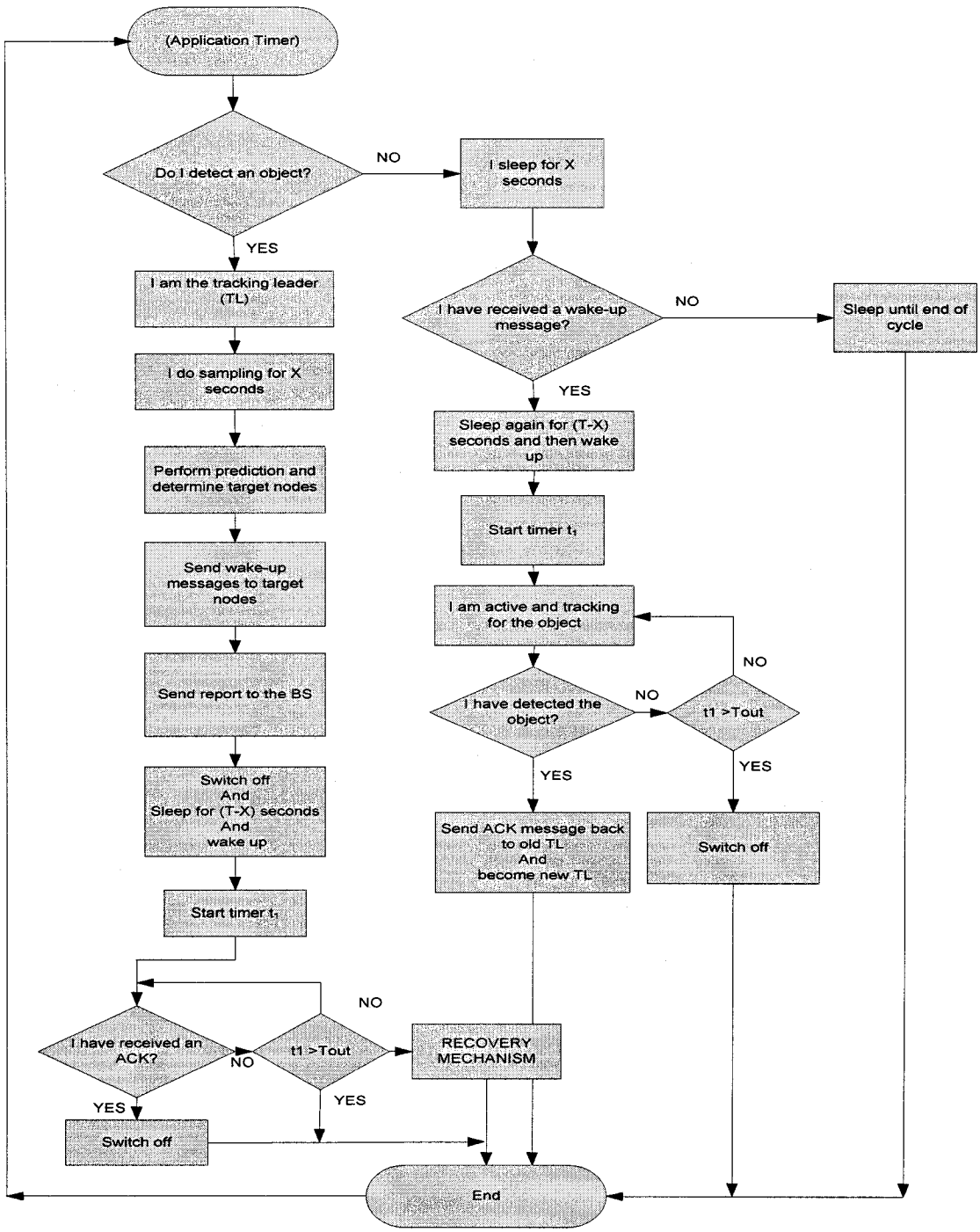


Figure 4.8 PES algorithm running in a node

After the initialization phase, where the data structures presented in section 4.2 are first set, the nodes wait to receive a message from the mobile object. This message is a broadcasted HELLO and will be detected by the closest cluster head.

Once the node has detected a message, it transitions to a new state that is managed in function *enterRcvpkt*, where the message's characteristics are analyzed. To find out which node sent the message, we use the QualNet API *MAPPING_GetNodeIdFromInterfaceAddress*. Messages between nodes are

- If the current node is the sink, the energy account is updated for this node due to the reception operation.
- If the message is a Wake Up one and was sent by the current target leader, the timers `WAITING_DURATION` and `ACTIVATION_DURATION` are triggered, in case we are in normal operation cycle or in a first stage of recovery, respectively. The node goes then to a sleep state by entering the function *enterNodeWaiting*. If the current state is second stage of recovery, a timer is started for the time length `SECOND_RECOVERY_DURATION` to effectively start tracking (no waiting time to start tracking in the second recovery phase). Afterwards, the node goes to the tracking state through the function *enterNodeTracking*.
- In case that the received message is an `ACKNOWLEDGE` one, if the node is an old current node that is waiting for confirmation that some other node has just detected the mobile object, the energy accountancy is updated first and then the node goes to sleep. This means also a mode switch (that consumes energy itself).
- For every node that detects the signal from the mobile object, it calculates its distance to the object and compares it to the remaining nodes' distance. If this is the closest node, it updates the corresponding field in the data structure of the node and this is the only detecting node for simulation purposes. It will be then responsible for tracking the object and sending the location information to the sink.

- If the current node is responsible to track the object, three situations can arise: i) the node is the new tracking leader and this is the first time the object is detected by the network, ii) the node is a tracking node and has detected the object, iii) the node is the current target leader and detects the object again in its detection area.
- In case this is the first detection of the object during the simulation, the sampling timer is started and the energies are updated.
- If the node is a tracking node and detects the object, it becomes the new target leader, it sends an acknowledge message to the old current node (the one that sent the original wake up message), and updates its energy metrics. It then starts sampling, by entering the function *enterNodeSampling*. The associated timer is started.
- If the node is the current target leader, the sampling timer is started again and the energy parameters are updated in consequence. The next sampling expiration event will take this point as a reference to calculate the time length during which the node was active and sampling the node, rather than the remaining duration of the detecting state duration.

The predicted position for the object according to PES algorithm is calculated by assuming a linear trajectory for the (T – X) interval (the final part of the operating cycle, after the sampling). The object stores its two last positions, and the detecting node uses this information to compute the next expected position. Vector `vector_two` contains the last two position's coordinates and vector `next_dest` stores the geographical coordinates for the next predicted location. This location is not a node position. We assume that the object's speed remains constant for that time. Figure 4.9 shows the calculation of the predicted position in cases where the coordinates x and y are different.

```
// next predicted destination
{
   next_dest[0].x =   vector_two[0].x +
speed_node*2*cos(atan((vector_two[0].y -
vector_two[1].y)/(vector_two[0].x - vector_two[1].x)));
   next_dest[0].y =   vector_two[0].y +
speed_node*2*cos(atan((vector_two[0].y -
vector_two[1].y)/(vector_two[0].x - vector_two[1].x)));
}
```

Figure 4.9 Calculation of the next expected location in PES

The next step in PES algorithm is to calculate the next predicted destination node, which will be the destination of the wake up message. Figure 4.10 shows the code to implement this functionality. The distances from the previously calculated destination point to each one of the nodes are calculated and the node with the least distance is chosen.

```
for (i = 2; i < node->numNodes; i++)
{
   MOBILITY_ReturnCoordinates(temp, &pos_node);
   destination_point.cartesian.x = next_dest[0].x;
   destination_point.cartesian.y = next_dest[0].y;
   COORD_CalcDistance(CARTESIAN,&pos_node, &destination_point,
&d);
   if (d< dmin)
   {
      dest_nodeId = temp->nodeId;
      dmin = d;
      closest = temp;
   }
   temp = temp->nextNodeData;
}
```

Figure 4.10 Calculation of the next expected node

After the calculation of the next predicted node that will host the object, the report to the sink is sent. The energy metrics are updated. Figure 4.11 shows the implementation of this functionality.

```
// the report is sent to to the sink
APP_UdpSendNewData(node,APP_FLOOD,dataPtr-
>clientAddr,(short)dataPtr-
>sourcePort,sinkNode,WakeUp,MAX_STRING_LENGTH,PROCESS_IMMEDIATELY,
TRACE_FLOOD);
node->BatteryEnergy += BATTERY_TX_CONSUMPTION;
node->txs ++;
```

Figure 4.11 Report to the sink

In case of recovery, the nodes neighboring the route must be woken up. Figure 4.12 shows this functionality.

```
// HEURISTIC ALL_NBR
Node *temp;
temp = (node->partitionData)->firstNode;
if (x_one == xD)
{
   for (i = 2; i < node->numNodes; i++)
   {
   MOBILITY_ReturnCoordinates(temp, &pos_node);
   x = pos_node.cartesian.x;
   y = pos_node.cartesian.y;
   if ((((x_one -1000)<= x) && (x <= (xD+1000)))&&((y_one <= y)
      && (y <= yD)))
   {
         recNode=MAPPING_GetDefaultInterfaceAddressFromNodeId
         (temp, temp->nodeId);
         if (node->nodeId != temp->nodeId && temp->nodeId != 1)
         {
            APP_UdpSendNewData(node,APP_FLOOD,dataPtr-
      >clientAddr,(short)dataPtr->sourcePort, recNode, WakeUp,
      MAX_STRING_LENGTH, PROCESS_IMMEDIATELY,TRACE_FLOOD);
            node->BatteryEnergy += BATTERY_TX_CONSUMPTION;

      }
   }
   temp = temp->nextNodeData;
}
```

Figure 4.12 Waking up neighbors to the route, ALL-NBR heuristic from PES

As to the GPBA protocol, first of all the profile fields are filled with the nodes' information. Figure 4.13 shows this functionality. The timeslots that limit the intervals of the profile are initialized first, followed by the frequent nodes themselves that are

expected to host the object during its movement. In the case of the figure, we have divided the day in 10 time slots and the movement is repetitive in a 105 seconds time frame.

```
//*********************************************************
// initializing the profile table
//*********************************************************
profile_table[0].time1 = 0;
profile_table[0].time2 = 15;
profile_table[1].time1 = 16;
profile_table[1].time2= 25

…
profile_table[9].time2 = 105;

for (int i=0; i<NUM_OF_INTERVALS;i++)
{
      for (int j=0; j<NUM_OF_NODES;j++)
   {
      profile_table[i].distribution[j]=0;
   }
}
profile_table[0].nodeId_1 = 2;
profile_table[0].nodeId_2 = 0; //7
profile_table[0].nodeId_3 = 0; // 3
…
for (int i=10; i<NUM_OF_INTERVALS;i++)
{
      profile_table[i].nodeId_1 = 0;
   profile_table[i].nodeId_2 = 0;
   profile_table[i].nodeId_3 = 0;
}
```

Figure 4.13 Initialization of the profile in GPBA

The information in the profile is used to predict the next destination node in GPBA, instead of the movement prediction which was performed in the case of PES. When the object is detected by a node, first this one determines if the object is following its profile or not. In case it is, the sampling period will not consume energy since the computation and sensing unit will be able to go to sleep. Figure 4.14 shows this check for an example time interval of the profile.

```
if (t <= 15)
{
      if ((profile_table[0].nodeId_1 == node->nodeId) ||
            (profile_table[0].nodeId_2 == node->nodeId) ||
            (profile_table[0].nodeId_3 == node->nodeId))
   node->inProfile = TRUE;
}
```

Figure 4.14 Determining if the object stays within its profile

Figure 4.15 outlines the process of waking up neighbors. The same function *WakeUpNeighbors* is used for the subsequent recovery phase, if it exists. If no nodes are found for that interval, the neighbors of the current node are woken up.

Figure 4.16 shows the determination of the next nodes to wake up, we first calculate the time slot corresponding to the present time plus (T-X) seconds and then look for nodes in that interval in the profile. More complex algorithms could have been implemented to optimize this choice real-time, but for the simulation purposes, tests with different amounts of nodes in the profile are also useful for our research.

```
void WakeUpNeighbors(Node *node, Node *node_profile,
                     FloodData * dataPtr, char *WUmsg)
{
   for (i = 1; i < node->numNodes; i++)
   {
     MOBILITY_ReturnCoordinates(temp, &pos_node);
     COORD_CalcDistance(CARTESIAN,&pos_node,
      &pos_nodeprofile, &d);
     if (d <= NEIGHBOR_DISTANCE && d> 0)
     {
       recNode=MAPPING_GetDefaultInterfaceAddressFromNodeId(temp
                                  ,temp->nodeId);
          if (node->nodeId != temp->nodeId && temp->nodeId != 1 &&
                        temp->wokenUp == FALSE)
          {
          APP_UdpSendNewData(node,APP_FLOOD,dataPtr-
          >clientAddr,(short)dataPtr->sourcePort, recNode, WUmsg,
          MAX_STRING_LENGTH, PROCESS_IMMEDIATELY,TRACE_FLOOD);
          temp->wokenUp = TRUE;
          node->BatteryEnergy += BATTERY_TX_CONSUMPTION;
          }
     }
     temp = temp->nextNodeData;
   }
```

Figure 4.15 Waking up the neighbor nodes in GPBA

```
double now = (double) getSimTime(node) / SECOND;
int t =  (int)now % 105;
int taux = t + APPLICATION_T - SAMPLING_DURATION;

if (taux <= 15)
{
      if (profile_table[0].nodeId_1 != 0 &&
profile_table[0].nodeId_1 != node->nodeId)
   {
            targetNode1=MAPPING_GetDefaultInterfaceAddressFromNode
            Id(node, profile_table[0].nodeId_1);
            APP_UdpSendNewData(node,APP_FLOOD,dataPtr->clientAddr,
            (short)dataPtr->sourcePort, targetNode1, WakeUp,
            MAX_STRING_LENGTH, PROCESS_IMMEDIATELY,TRACE_FLOOD);
      node->BatteryEnergy += BATTERY_TX_CONSUMPTION;
      node->txs ++;
   }
   else if (profile_table[0].nodeId_1 != node->nodeId)
   WakeUpNeighbors(node, node, dataPtr, WakeUp);

      if (profile_table[0].nodeId_2 != 0 &&
      profile_table[0].nodeId_2 != node->nodeId)
   {
      (same logic)

      }
      if (profile_table[0].nodeId_3 != 0 &&
profile_table[0].nodeId_3 != node->nodeId )
      {
         (same logic)
      }
   }
```

Figure 4.16 Search of target nodes to wake up according to profile in GPBA

Figure 4.17 shows the algorithm to be executed at every node to run the GPBA algorithm.

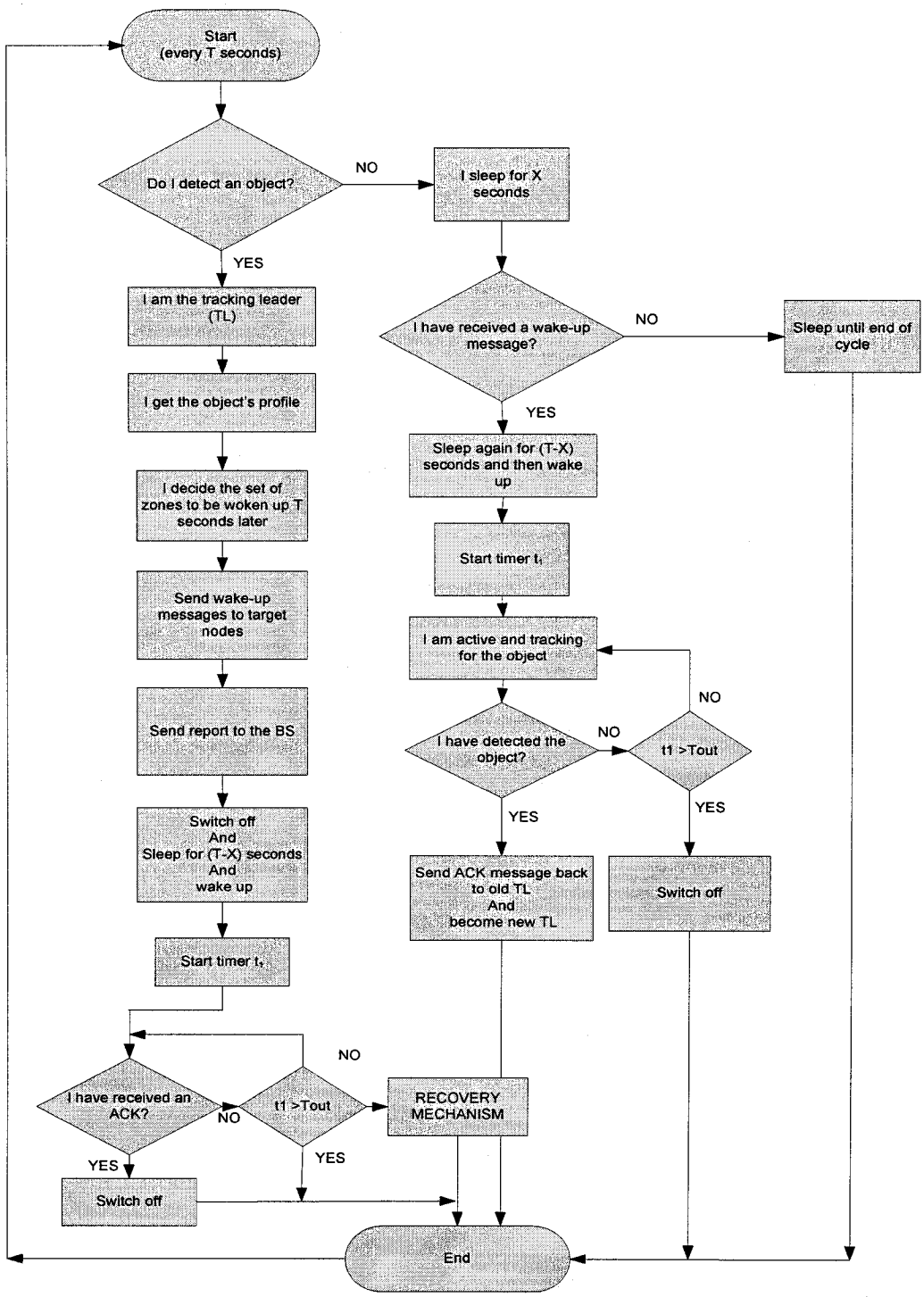


Figure 4.17 GPBA algorithm running in a node

4.5 Experiments and simulation plan

This section presents the experiments plan to be used to evaluate the performance of GPBA, the profile-based energy-saving algorithm for OTSNs proposed in this report. Besides the scenarios' configuration and the simulation parameters, the experiments' plan determines a number of performance metrics that will be used to compare the proposed method to other existing algorithms.

4.5.1 Configuration of the simulation

In our simulations, we consider a wireless sensor network with 30 nodes uniformly distributed, one mobile object and one sink. The monitored region that contains all those nodes is 6000 × 6000m. In order to test the scalability of our protocol, we will vary the number of nodes on the route followed by the mobile object to 30, 40, 60 and 80. All nodes have a wireless communication interface of type 802.11b in ad hoc mode. This MAC layer protocol enables the nodes to transmit point-to-point without centralized access point and communicate in broadcast mode.

The mobility model is implemented through configuration files manually designed to define a customized trajectory. This allows us to use pre-defined profiles with several accuracies to test the proposed algorithm, since the profile update (learning) functionality has not been implemented (out of the scope of our work). We assume therefore that the network has already learnt the object's behavior before start the simulation, and the profile exists.

One of the nodes plays the role of mobile object. It is the only mobile object in the network and the rest of nodes track it. Another node is the sink and the location information reports are sent to it.

Several trajectories have been used for the simulation of the mobile object.

Table 4.1 summarizes the configuration of the parameters used in the simulation. The changing parameters are highlighted in grey.

Table 4.1 Configuration of the simulation parameters

Parameter	Value
Size of the simulation area (m)	6000 × 6000
Number of nodes on the object's route	20, 30, 60, 80
Nodes' distribution	Uniform
MAC layer protocol	802.11b
Throughput (Mbps)	2
Antenna type	Omnidirectional
Transmission range (m)	50
Propagation model	Free space
Mobility model	FILE
Profile's probability to host the object	60%, 100%

4.5.2 Performance metrics

In order to compare our algorithm's performance, we have used the following six metrics:

- *Energy consumption* : the total energy consumed by all the sensor nodes in the network (excluding the mobile object and sink) ;
- *Missing rate* : number of object missing relative to the number of reports that the sink should have received in the simulation time;
- *Activation Time* : time that the nodes are in active mode, sensing and tracking;
- *Number of Transmissions, Receptions* : total in the network including the sink ;

We will study the influence of these factors in the working of our protocol:

- The profile's accuracy.
- The number of nodes in the profile;

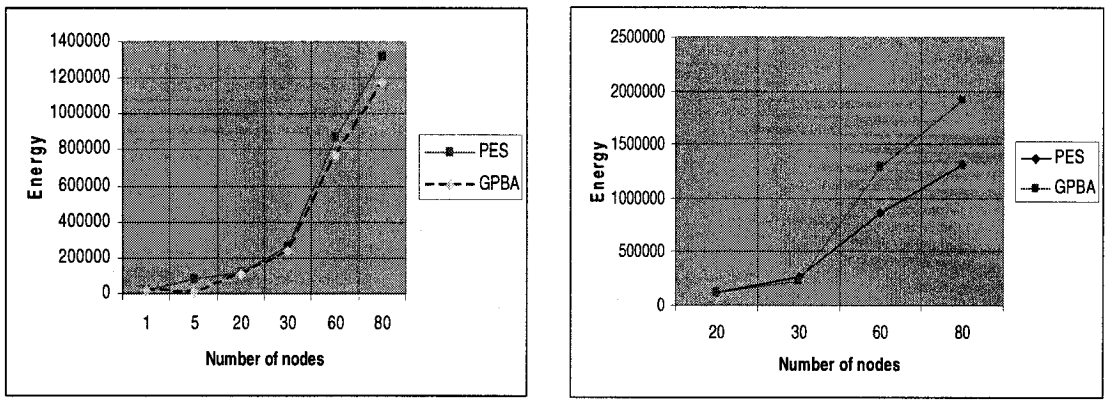
4.6 Analysis of the simulation results

We compare the obtained results to the protocol PES. The results for the Naive protocol are taken as a bottom line reference and are not calculated, since no simulation is necessary for that, given a number of nodes, their characteristics and the reporting

period. Naïve protocol does not perform any energy tuning and its results are forcedly costlier in energy than PES.

4.6.1 Global energy consumption in the network

Figures 4.18 (a) and (b) show the energy consumption as a function of the number of nodes along the mobile object's trajectory when the object follows its predicted behavior (the zones contained in the profile) 100% of the time and 60% of the time. The number of nodes in the profile is 1 in both cases.



(a) 100% predictability **(b) 60% predictability**

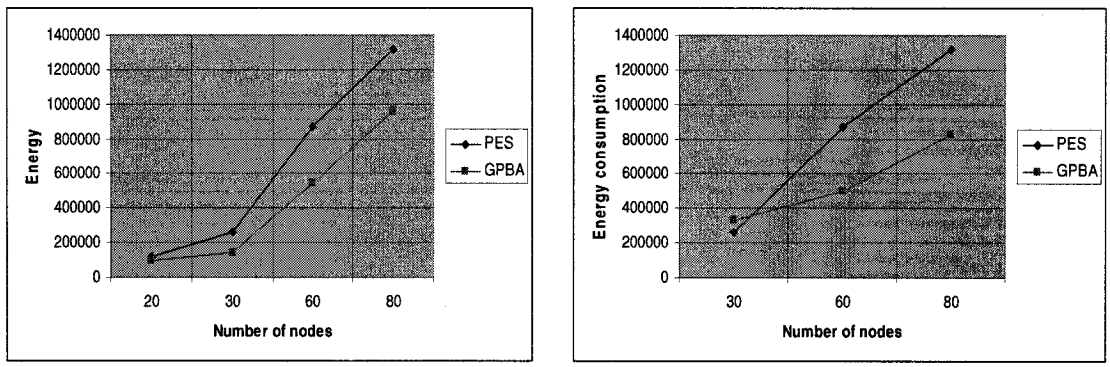
Figure 4.18 Energy consumption in the network a with a 1 node profile

The fact of having 100% predictability does not mean that the object will always tracked at the first try. Actually, it means that there is always one zone in which the mobile object should be if there were no time de-synchronizations. In effect, the time cycle is not always the same, but the profile's time slots have fixed limits in our implementation. This means that sometimes, none of the nodes that have been woken up for being in the profile is going to effectively host the object, resulting in an object missing and a recovery mechanism taking place.

We observe in Figure 4.18 (a) that GPBA saves an average of 10% energy as compared to PES. The improvement is better for the last simulation scenario, with 80 nodes along the route. Regarding Figure 4.18 (b), the energy consumed by GPBA is 20% costlier than PES in average, and is costlier when more nodes are on the route. This difference in the performance of GPBA can be explained by the fact that GPBA

consumes more energy when the predictability is lower. For the same number of nodes in the profile, the communication cost for inter-nodes wake up messages is the same as in PES. But less predictability means more recovery mechanisms and thus, more communication and activation time than in PES. Clearly the predictability floor for the object's movement is situated between 60% and 100%.

Figures 4.19 (a) and (b) show the global energy consumption as a function of the number of nodes along the trajectory of the mobile object when there are 3 nodes in the profile and their cumulative probability to host the object is 100% and 60%. GPBA always outperforms PES.



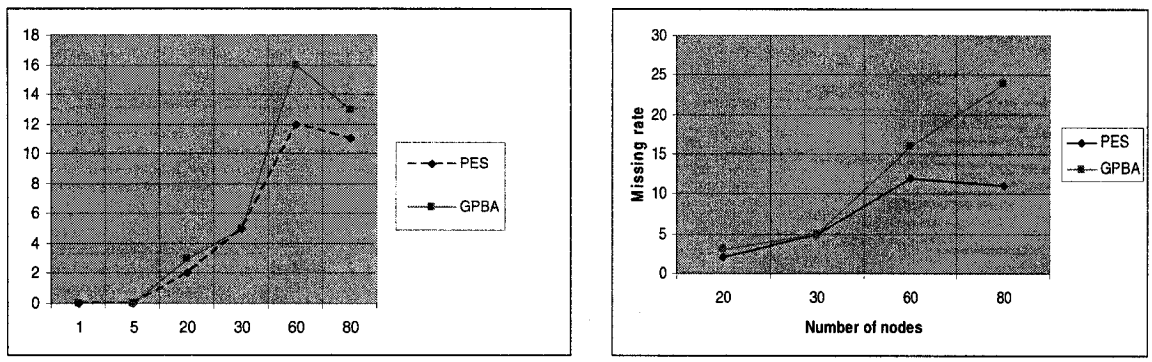
(a) predictability = 100% **(b) predictability = 60%**

Figure 4.19 Energy consumption in the network with a 3 nodes profile

We can observe in Figure 4.19 (a) that GPBA consumes less energy than PES in all the cases. The average improvement is 30%. When predictability is lower, the energy consumption does not change much for GPBA. This can be explained by the fact that the communication costs are practically the same as in a) and the importance of the extra cost caused by the recovery mechanisms is overwhelmed by the increase in communication cost.

4.6.2 Object's missing rate

Figures 4.20 (a) and (b) show the object missing rates when there is 1 node in the profile and their predictability, 60% or 100%. GPBA has a higher missing rate than PES.



(a) predictability =100% **(b) predictability = 60%**

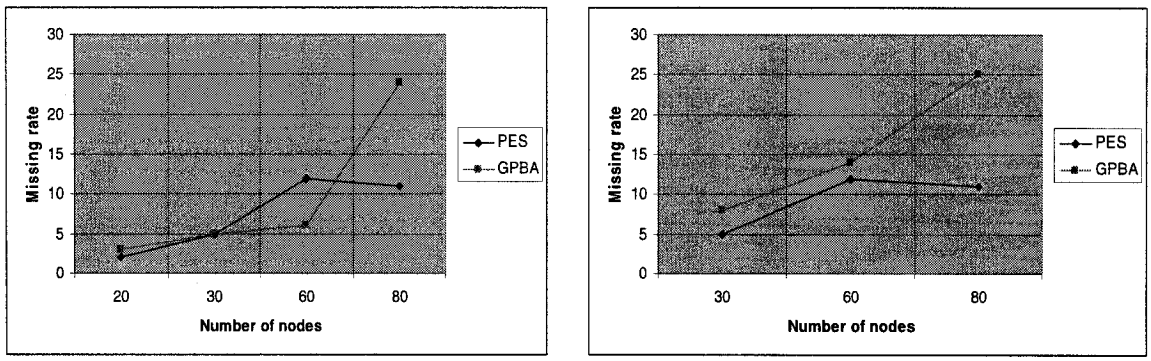
Figure 4.20 Object missing rate in the network with a 1 node profile

In figure 4.20 (a), we see that the missing rate for GPBA is always higher than in PES and the difference increases with the number of nodes on the trajectory. The information contained in the profile appears to be less reliable than the prediction. This can be explained by the fact that the movement is easy to predict by PES, whereas GPBA has a higher probability to miss the object because the profile is static during the simulation and cannot react to the time de-synchronizations that cause the object missing. In Figure 4.20 (b), the missing rate is also higher for GPBA than for PES. Less predictability for only node means more probability to miss the object for GPBA. The average increase in missing rate is 60% for 60% predictability and 23% for 100% predictability.

Figures 4.21 (a) and (b) show the object missing rates when there are 3 nodes in the profile and their predictability is 60% or 100%.

With more nodes in the profile the tendency of GPBA relative to PES changes. Like in figures 4.21 a) and b), these figures show an increase in the missing rate when the object passes by more nodes. The average missing rate increases are 68% for 60%

predictability (60% for 1 node) and 27% for 100% predictability (23% for one node in the profile). The influence of the probability is more significant than the number of nodes in the profile.

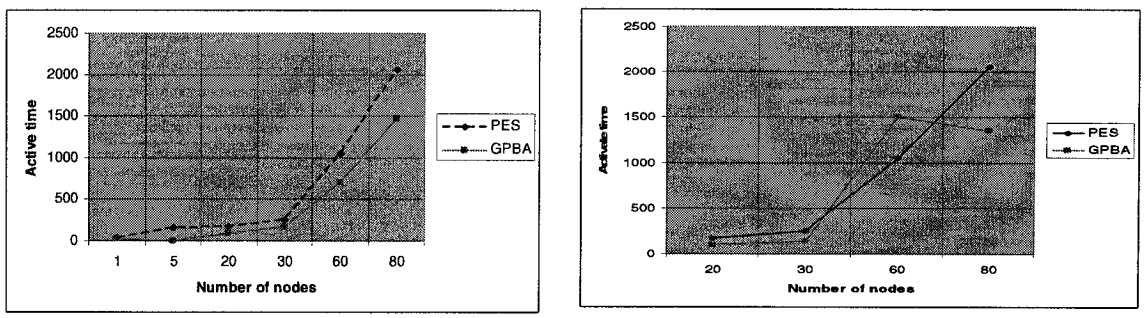


(a) predictability =100% **(b) predictability =60%**

Figure 4.21 Object missing rate in the network with a 3 nodes profile

4.6.3 Activate time

Figures 4.22 (a) and (b) show the cumulative time during which the nodes' sensing and computation units have been in active mode when there is 1 node in the profile and with predictabilities 60% or 100%. GPBA achieves shorter activation times than PES in almost all simulation scenarios.



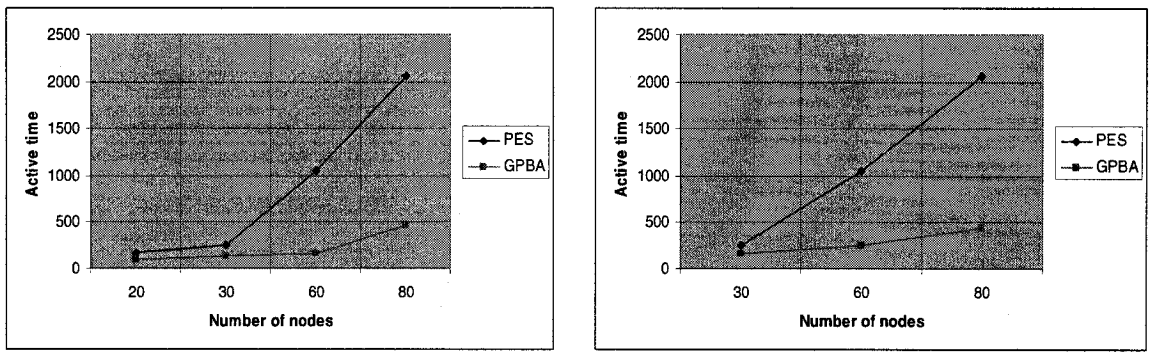
(a) predictability =100% **(b) predictability =60%**

Figure 4.22 Activate time in the network with a 1 node profile

Time in the active state consumes sensing energy. GPBA main source of improvement regarding to PES is its history-based knowledge in-network, which makes it unnecessary to sample the object's movement if this one is moving within its profile.

Results from simulation confirm this intuition: when only 1 node is the profile, the activate time is 70% of the PES active time in average, when the node in the profile has a 100% presence along the object's trajectory. When the profile cumulates 60% predictability instead, the ratio is 80%, slightly less efficient than for the 100% case.

Figures 4.23 (a) and (b) show the activate times when there are 3 nodes in the profile and their predictability is 60% or 100%. We observe that GPBA yields better (lower) activate times than PES for both scenarios.



(a) predictability =100% **(b) predictability =60%**

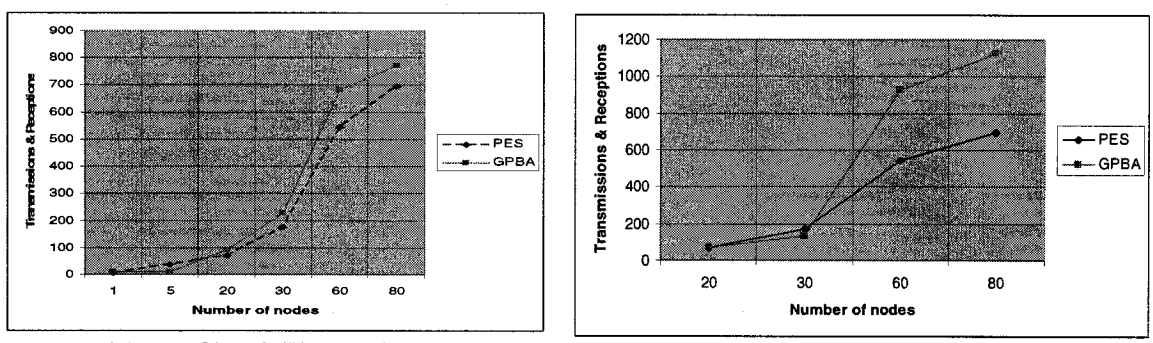
Figure 4.23 Activate time in the network with a 3 nodes profile

In Figure 4.23 a) we find that GPBA succeeds to reduce the activate time an average of 65% relative to PES. The reduction is almost the same for the case of Figure 4.17 b), when the predictability is lower. If we compare these figures to 4.22 a) and b), we observe a gain of 30% in activate time reduction when the predictability is 100% and more than 40$ for 60% predictability for the increase in number of nodes in the profile. This is surely due to the fact, explained before, that the number of nodes in the profile is relatively more important than the probability that the object moves within those nodes in theory. Due to the way the simulation is designed, the more nodes we have in the profile, the most likely they are to host the object, even if they are not present all along the object's trajectory. This appears truer when we realize that the nodes in the profile are normally neighbors, so their capacity to host the object and reduce the missing rate is

higher. The recovery processes increase the activate time (for the nodes that perform recovery must be active to track the object).

4.6.3 Number of transmissions and receptions at nodes

Figures 4.24 (a) and (b) show the number of transmissions and receptions at all nodes in the network, excluding the mobile object, when there is 1 node in the profile and with predictabilities 60% or 100%. GPBA yields a higher number of transmissions and receptions than PES for all the situations.

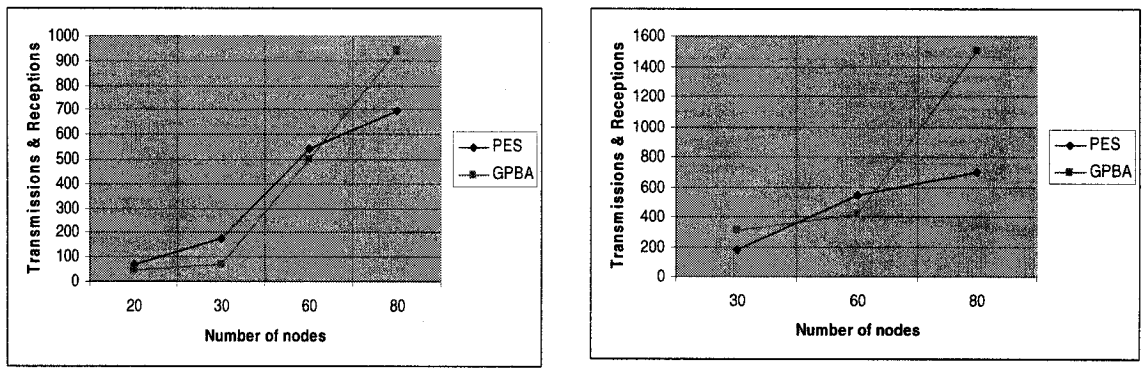


(a) predictability =100% **(b) predictability =60%**

Figure 4.24 Transmissions and receptions in the network with a 1 node profile

GPBA is prone to generate more transmissions and receptions than PES, with two factors determining that this number of communications overwhelms the savings achieved in sensing: the number of nodes in the profile and the probability that these nodes hold. Even though the number of transmissions/receptions will increase with the number of nodes in the profile (these nodes will have to be woken up), the number of transmissions to the sink will likely reduce if the profile does effectively host the object. Figure 4.24 a) shows that GPBA steadily yields more transmissions and receptions at the nodes than PES, 6% in average, whereas this ratio increases to 26% when the predictability is only 60%. We can explain this increase by the number of nodes that have to be woken up for the recovery mechanism, which is a factor that does not have any influence in PES' performance, since obviously this one does not change with the profile's predictability.

Figures 4.25 (a) and (b) show the number of transmissions and receptions in the network when there are 3 nodes in the profile and their predictability is 60% or 100%. GPBA shows more variability in its number of transmissions and receptions relative to PES. There is not a clear "winner" between the two algorithms according to these figures.



(a) predictability =100% **(b) predictability =60%**

Figure 4.25 Transmissions and receptions in the network with a 3 nodes profile

In Figure 4.25 a), we observe that GPBA obtains less transmissions and receptions than PES for shorter object's routes. In average, GPBA has 85% of the number of transmissions and receptions of PES. Longer routes mean more probability of object missing and in that case, more transmissions and receptions are performed to wake up the neighboring nodes to the ones that are already in the profile. In Figure 4.25 b) it can be observed that the number of communications in GPBA is higher than in PES; the predictability here is lower and the missing rate is higher than in PES comparatively. This explains the increase in transmissions and receptions.

We can see, comparing both figures 4.24 and 4.25, that the best scenario for the number of communications is having more nodes in the profile with the maximum profile probability: this scenario is even better than having fewer nodes in the profile with the same cumulative probability (85% to 106% of the PES number of communications).

4.6.7 Synthesis of the results

Once the analysis is done, we will now show the merits of our proposed algorithm. Table 4.2 and Table 4.3 outline the average results for the comparison GPBA/PES of the metrics that were defined previously.

Table 4.2 Summary of average simulation results with 1 node profile

	Energy Consumption	Missing rate	Active time	Transmissions And Receptions
60% cumulative probability	121%	160%	80%	126%
100% cumulative probability	76%	123%	65%	106%

Table 4.3 Summary of average simulation results with 3 nodes profile

	Energy Consumption	Missing rate	Active time	Transmissions And Receptions
60% cumulative probability	81%	168%	36%	156%
100% cumulative probability	68%	127%	35%	85%

The results show that the relative performance of GPBA to PES is interesting. The energy consumption is improved a 20-30% when we have 3 nodes in the profile, and 20% with only one node that is present 100% of the time along the object's trajectory. It only performs worse than PES for one node in the profile having being present 60% of the time the object is moving. These results show that the algorithm can also be advantageous if we consider the additional energy consumption for the mobile object's electronic tag, transmissions to the nodes in the network as well as the extra computation cost that has not been considered in the simulation.

The missing rate is less advantageous in our proposed algorithm than in the prediction-based one. Especially when the profile only holds 60% of the times, the number of missing reports that fail to arrive on time to the sink increases relative to PES. The object is always tracked eventually, but we pay a price in QoS for our energy savings. However, the case for which we obtain better QoS is also the case for the

highest energy savings, especially when the nodes are present in the object's route all along its trajectory (routine behavior).

The reduction in the active time is the main source of energy savings for GPBA as we can see in the tables. The savings can reach 70% and their importance is much more significant than the increase in the number of transmissions and receptions. The number of transmissions and receptions is further reduced when the reduction in the active time is more significant and explains the greater energy savings for the case in which more nodes are present in the profile, more frequently on the object's route.

To sum up, the performed simulations help to highlight the merits of our approach to reduce the energy consumption in OTSNs by utilizing the information on the historical behavior of mobile entities which have a behavior that could be considered predictable enough.

CHAPTER V

CONCLUSION

In this thesis, we have proposed an algorithm to reduce the energy consumption in sensing, computation and communication in object tracking wireless sensor networks. We assume that the mobile objects tracked by the network have memory resources that cooperate with the sensors themselves to build and transmit the information about the regularity of their own movements. Compared to a prediction-based algorithm taken as reference, the simulation results show a significant energy reduction.

This final chapter shows a summary of our work and its main contributions. In the following, we discuss some of the limitations of our work and provide some hints for future investigations on the subject.

5.1 Summary of the work and originality of our contributions

The emergence of wireless sensor networks, coupled with the constraints that are associated to their nodes, will demand great challenges in energy consumption. The growing presence of context aware networks calls for the utilization of behaviors' profiles for new services that can react to the users and personalize their functionalities, recognizing their preferences.

One of the most promising applications for sensor networks is object tracking. The predictable behavior of the tracked objects can be utilized to:

1) Learn the object's behavior
2) Save energy in the sensing and reporting operations, while signaling the irregularities that may arise in the object's trajectory.

The proposed algorithm uses the mobile object's capabilities to build its profile while it passes by the nodes of the network; if the object is inside its profile, sensing and

computation units can go to sleep and save reports to the sink when the profile is followed. This kind of behavior is not expected to exist in all cases and applications, and the proposed scheme would be especially interesting in the cases where the behavior is predictable enough to produce gains.

We have tested and compared our algorithm to a prediction-based one in which the future position of the object is calculated in a continuous way, with no learning and sending reports to the sink periodically. A basic algorithm, the naïve one in which all the nodes are active and tracking all the operation time, is not simulated since its performance is useful only as a reference. Energy is not tunable in this naïve approach.

Our proposed algorithm replaces the prediction computations with the profile's construction and reduces sensing time as well as communications, provided that the object follows its own routine to an extent. The nodes that will be activated for being in the profile are static in our simulations but could well change dynamically in response to the current object's behavior. The set of nodes that are woken up even within the profile can also change dynamically depending on their changing probabilities to detect the object. If the object does not follow its historical behavior, the nodes which are woken up in the profile will not detect the object and there are chances that more communication costs will be spent than in a prediction based scheme. This scenario is not sure however since the prediction has also a degree of inaccuracy and more than one node (the destination) is woken up by the prediction-based approach: also the nodes on the route or neighboring the route might be woken up to avoid recovery processes.

We have also proposed a modification to the recovery algorithm used in the prediction-based scheme. Instead of waking up the nodes on the route to the predicted destination, which depends on the current speed and direction of the object, we wake up the nodes neighboring that route. This aims to be coherent with a realistic mobility model such as Gauss-Markov, which assumes that the animals or persons have a destination in mind during their movement, so the final destination and its surroundings appear more likely to host the object than the route to reach it.

Our work outlines that 10% to 30% savings can be achieved by our scheme relative to a prediction-based algorithm. If more nodes are present in the profile or more probability is stored in it, the savings are even higher. The activation time is reduced a 20% even with a predictability as low as 60% with only node in the profile. The missing rate also increases relative to the prediction-based approach, but to a smaller extent. The savings in sensing and computation time overweigh the increase in transmissions and receptions.

5.2 Limitations of our work

Even though the performance analysis of our proposal yields advantageous results as compared to the reference, some limitations exist in our work. The first limitation is the abstraction from the details of the learning mechanism. In effect, while several techniques are available to predict the object's movement and make the network itself learn the behavior of the object, no specific approach is considered in our work, that assumes the pre-existence of a profile in the nodes. The convenience of one or another learning method is important due to the especial energy constraints in the nodes. It is likely that not all the algorithms will be equally suitable for this task. The learning mechanism operation will have an influence in one of the operations of the approach, when the object is detected, the profile will have to be updated and the duration of this operation must be coherent with the rest of activities performed at the node. This requirement is even more important when we think of the periodic reports that have to be sent to the sink as required by the application.

A second limitation lies in the trajectories employed for the simulation, though it is linked to the previous one. We have considered routine movements that are regularly followed by the tracked object, normally straight lines or rectangles, with punctual detours from the "predicted" trajectories. We have used file-based mobility in QualNet. The Random-walk model is not suitable to our research, since our algorithm assumes a certain routine in the object's behavior and a random movement could not be exploited by a learning network.

The composition of the profile is static and is not updated with the object's movement. A maximum of three nodes are considered for each time slot, while a probability-based criteria would be an alternative. This would be feasible if we used a dynamic learning-based approach to update the profile, which is not the case as we already discussed.

The proposed method assumes that the mobile object must have electronic resources to compute its own profile and detect the network to transmit this profile to nearby nodes. This requirement is not always possible or convenient, depending on the nature of the object or animal. If the tracked entity already carries electronic equipment, this limitation would be attenuated, but it can be a limiting factor for other situations.

5.3 Future work

One possible direction for future work is an algorithm to choose the optimal set of nodes to be woken up. The tracking nodes could run a simple algorithm to balance the communication costs to wake up target nodes and their sensing and computation costs, altogether with the probability of recovery. Waking up more nodes means spending more energy in transmissions and sensing time, but also lowers the chances of spending even more energy in recovery mechanisms and worsening the QoS due to the late report to the sink. Another possible future direction is the learning mechanism to be run online at the nodes, be it based on Bayesian networks, or neural networks and Kalman filters. The choice of one or another must take into account the limited resources of nodes and the application requirements regarding timely reports to the sink.

The profile information which is stored by the network can be further utilized to optimize other aspects of the sensor network (cross-layer optimization). It could potentially be used for routing, MAC or clustering schemes that could be more energy-efficient than non-profile based ones.

Finally, it would be worth to further investigate an extension of the proposed approach to several objects that would move in the monitored area. The resulting

parallel processes that should be executed by the nodes and the possible overlapping situations could be analyzed more in depth.

REFERENCES

[1] I.F.Akyildiz, I.H. Kasimoglu, "Wireless sensor and actors networks: research challenges", Ad *Hoc Networks,* Vol. 2, No. 4, pp. 351-367, 2004.

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a survey", Computer *Networks*, Vol. 38, No. 4, pp. 393-422, 2002.

[3] P. Beyens, A. Nowe, K. Steenhaut, "High-density wireless sensor networks: a new clustering approach for prediction-based monitoring", *Proceedings of the Second European Workshop on Wireless Sensor Networks*, pp. 188 – 196, 2005.

[4] E. Cayirci, I.F. Akyildiz, "User Mobility Pattern Scheme for Location Update and Paging in Wireless Systems", *IEEE Transactions on Mobile Computing*, Vol. 1, No. 3, pp. 236-247, 2002.

[5] E. Elnahrawy and B. Nath, "Context-Aware Sensors", *Wireless Sensor Networks*: *First EuropeanWorkshop, EWSN 2004,* pp. 77 – 93, 2004.

[6] M. Gaynor, S. Moulton, M. Welsh, E. LaCombe, A. Rowan, J. Wynne, "Integrating Wireless sensor networks with the grid", *IEEE Internet Computing*, pp. 32-39, 2004.

[7] B. Gerkey, M. Mataric, "A market-based formulation of sensor-actuator network coordination", *Proceedings of the AAAI Spring Symposium on Intelligent Embedded and Distributed Systems,* pp. 21-26, Palo Alto, California, March 25-27, 2002.

[8] S. Goel, A. Passarella and T. Imielinski, "Using buddies to live longer in a boring world", *Technical Report* DCS-TR-558, Rutgers Department of Computer Science, 2004.

[9] Z. Guo, M. Zhou, "Prediction-based object tracking algorithm with load balance for wireless sensor networks", *IEEE International Conference on Networking, Sensing and Control*, pp. 756-60, Tucson, Arizona, March 19-22, 2005.

[10] Martin Haenggi, "Mobile sensor-actuator networks: opportunities and challenges", *Proceedings of the 7th IEEE International Workshop on Cellular Neural Networks and Their Applications*, pp. 283- 290, Frankfurt, Germany, July 22-24, 2002.

[11] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocols for wireless microsensor networks", *Proceedings of the 33rd International Conference on System Sciences (HICSS '00)*, Vol.8, pp. 8020-8029, Hawaii, January 4-7, 2000.

[12] C. Hsin, M. Liu, "Network Coverage Using Low Duty-Cycled Sensors: Random & Coordinated Sleep Algorithms", *3rd ACM International Symposium on Information Processing in Sensor Networks, IPSN'04*, pp. 433-442, Berkeley, California, April 26-27, 2004.

[13] W. Hu, N. Bulusu, S. Jha, "A communication paradigm for hybrid sensor/actuator networks", *15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Vol. 1, pp. 201- 205, Barcelona, Spain, September 5-8, 2004.

[14] C. Intanagowiwat, R. Govindan, D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks", *Sixth Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pp. 56-67, Boston, Massachusetts, August 6-11, 2000.

[15] A. Mahapatra, K. Anand, D.P. Agrawal, "QoS and energy aware routing for real-time traffic in wireless sensor networks", *Computer Communications*, Vol. 29, No. 4, pp. 437-445, 2006

[16] S. Simic, "A Learning-Theory Approach to Sensor Networks", *IEEE Pervasive Computing,* Vol. 2, No. 4, pp. 44-49, 2003.

[17] A. Sinha, A. Chandrakasan, "Dynamic Power Management in Wireless Sensor Networks", *IEEE Design and Test of Computers*, Vol. 18, No. 2, pp. 62-74, 2001.

[18] C. Schurgers, V Tsiatsis, MB Srivastava, "STEM: Topology Management for Energy Efficient Sensor Networks", *IEEE Aerospace Conference Proceedings*, pp. 3-1099 – 3-1108, Big Sky, Montana, March 9-16, 2002.

[19] I. Stojmenovic, *Handbook of Sensor Networks: Algorithms and Architectures*, Wiley, Hoboken, New Jersey, 2005.

[20] J. Tan, N. Xi, "Integration of sensing, computation, communication and cooperation for distributed mobile sensor networks", *Proceedings of the 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, pp.54-59, Changsha, Hunan, China, 2003.

[21] H. Yang, B. Silkdar, "A protocol for tracking mobile targets using sensor networks", *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 71 – 81, Anchorage, Alaska, 2003.

[22] G. Wan and E. Lin, "A dynamic paging scheme for wireless communication systems", *Proceedings of the Third Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 195-203, Budapest, Hungary, 1997.

[23] K. Wu, J. Harms, E.S. Elmallah, "Profile-based protocols in wireless mobile ad hoc networks", *Proceedings of the 26th Annual IEEE Conference on Local Computer Networks*, pp. 568-575, Tampa, Florida, 2001.

[24] V. Raghunathan, C. Schurgers, S. Park, M.B. Srivastava, "Energy-Aware Wireless Microsensor Networks", *IEEE Signal Processing Magazine*, Vol. 19, No. 2, pp. 40-50, 2002.

[25] Y. Xu, J. Winter, W.-C. Lee, "Prediction-based Strategies for Energy Saving in Object Tracking Sensor Networks", *Proceedings of the 2004 IEEE International Conference on Mobile Data Management (MDM'04)*, pp. 346-357, Berkeley, California, January 19-22, 2004.

[26] F. Zhao, L. Guibas, *Wireless Sensor Networks: An information processing approach*, Morgan Kaufmann, San Francisco, 2004.

APPENDIX

RESULTS OF ANALYTICAL TESTS OF THE PROPOSED ALGORITHM GPBA

Performance comparison for scenario 1

If the object is within its profile, no report to the sink is needed and there is no need for a sampling time to capture the movement's characteristics. The larger E_{on} and E_{chsink}, the larger the relative improvement over PES brought by GPBA should be.

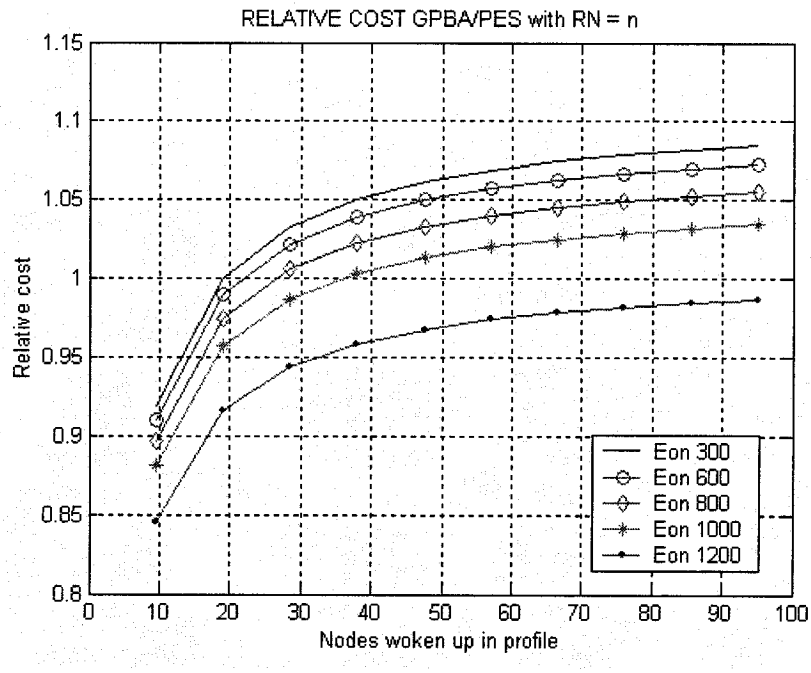


Figure A.26 GPBA/PES for scenario 1 when the activation cost varies

If we set the number of nodes on the route in PES (RN) equal to the number of activated nodes in the profile (n), as in Figure A.26, the inter-cluster heads communication costs are balanced and the difference in cost results from the sensing

part and the communication to the sink. Figure A.26 shows that, the larger the energy consumption in sensing, the larger the gain of GPBA relative to PES. We notice also that an increase in the number of active nodes means that PES approaches GPBA's performance, due to their common inter-nodes communication costs, which outweighs the sensing costs.

In a real scenario, we do not know *a priori* the relationship between RN and *n*; the value of RN will depend on the nodes' density and the speed of the object, whereas the value of *n* will also depend on the nodes' density, but mainly on the randomness in the object's behavior. The following test showcases this; we can notice in Figure A.27 that whenever RN is larger than *n*, GPBA outperforms PES. The flat shape of the curves when *n* approaches *S* is indicating that the performance scales well with the size of the profile. GPBA's performance improves significantly only when the number of nodes in the profile is less than 20% of the total nodes in the network. This condition should be met for a scheme that takes advantage of a predictable behavior and from movement's patterns.

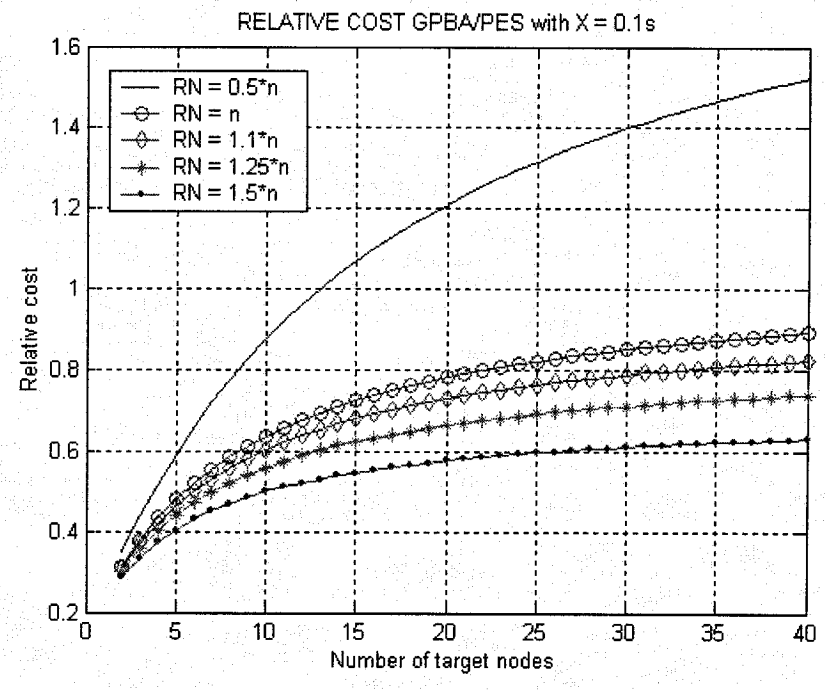


Figure A.27 GPBA/PES for scenario 1 when the ratio RN/n changes

A possible source of additional cost in the case of GPBA is the communication between the mobile object's electronic tag and the neighboring cluster head. The parameter that models this is E_{chobj}. In Figure A.28 we observe that it takes E_{chobj} = 500 mW when 10 nodes are woken up in order for GPBA to be less efficient than PES.

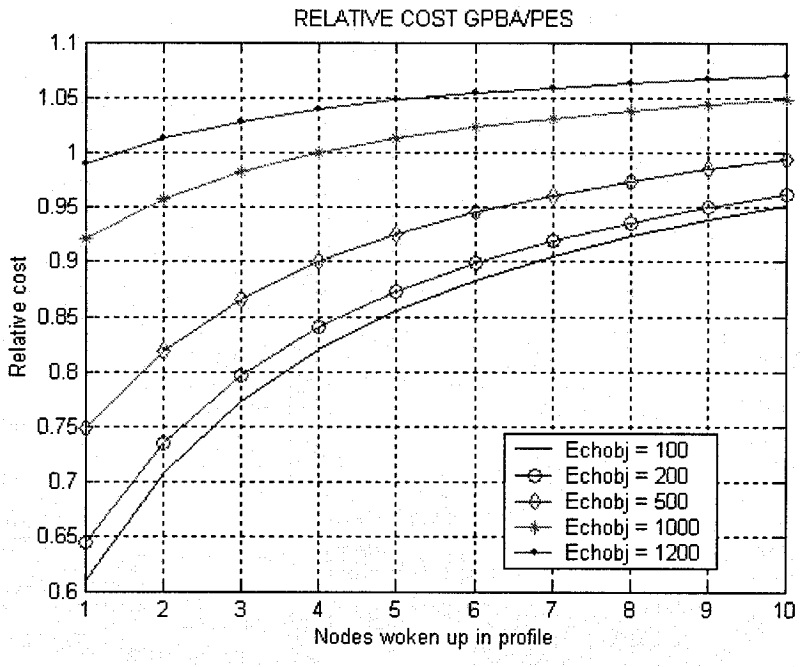


Figure A.28 GPBA/PES for scenario 1 with varying E_{chobj}

Performance comparison for scenario 3

In the first stage of recovery, the cost is driven by the number of waked up nodes; GPBA performs better than PES whenever the number of nodes on the route from the origin to the destination in PES is twice the number of the nodes activated in the profile or higher, as it can be observed in Figure A.29. This result was expected since the total number of active nodes is greater in GPBA than in PES.

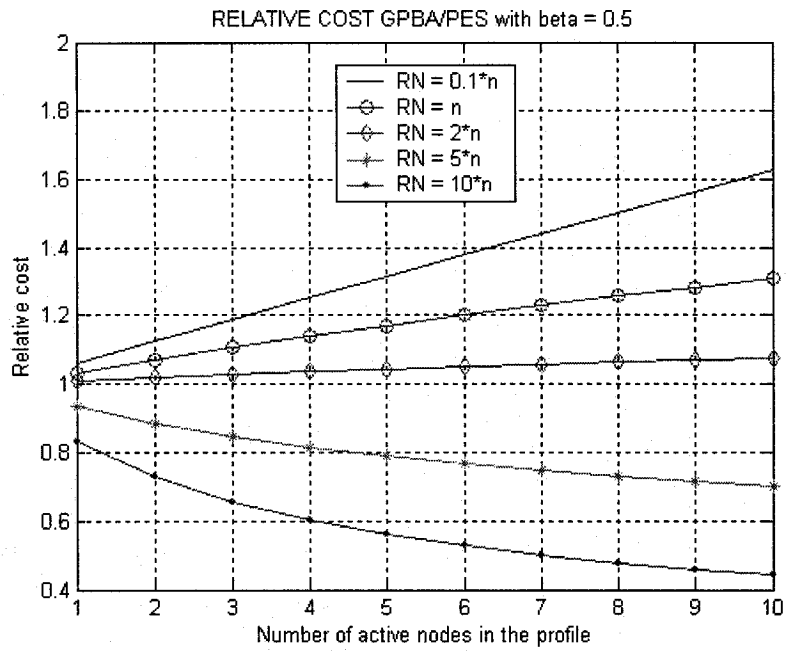


Figure A.29 GPBA/PES for scenario 3 with varying n with β = 0.5

If the probability of success for the first stage of recovery improves to 0.9 for the two approaches, PES becomes even more efficient than GPBA, getting to 25% less cost when RN = n =10 (Figure A.30). This is because the relative weight of the first stage of recovery gets more important, and PES is more efficient in that stage.

The moving behavior of the mobile object is modeled by the Gauss-Markov mobility model [25]. This model assumes that mobile entities usually travel with a destination in mind, so mobile's location and velocity in the future are likely to be correlated with its current location and velocity. The random-walker model would be unsuitable however, due to its memory-less nature. PES predicts the future location of the object based on the probability density function of the mobile's location. The distance between the mobile's location and the last reported location follows a Gaussian distribution. Most of the probability is then concentrated around the mean, which is the predicted location.

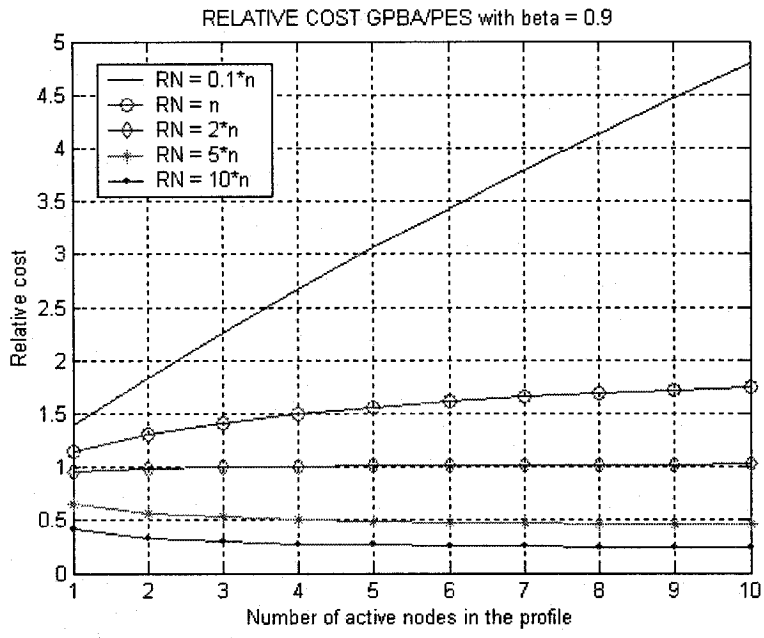


Figure A.30 GPBA/PES for scenario 3 with varying n. β = 0.9

β is the probability that the first stage of recovery succeeds in tracking the object. Due to the Gauss-Markov mobility model, the profile-based methods that wake up the zones situated around the predicted destinations (activated in the profile) are supposed to be more accurate than PES. This parameter can thus improve the relative performance of GPBA and LPBA, since the second stage of the recovery will happen less often than it does in PES. Figure A.31 confirms that, when β of GPBA becomes 10% better than β of PES, the relative cost for GPBA is reduced 5%. But the number of nodes in the profile is a more important factor, since more than 6 nodes give a better performance to PES even if the probability of success in the first recovery was 40% better in GPBA. The advantage of PES would still be valid even if the communications costs were lower than the sensing costs.

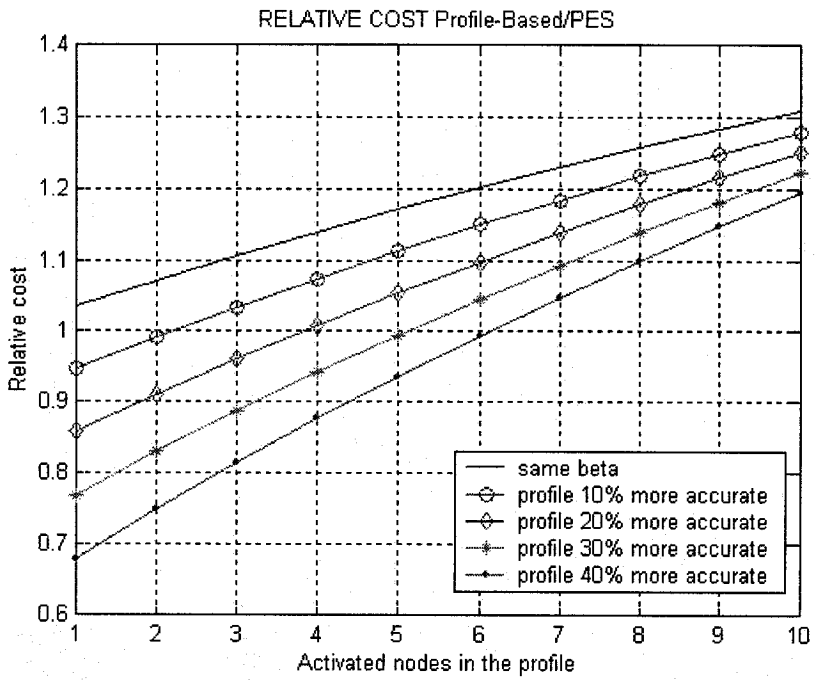


Figure A.31 GPBA/PES for scenario 3 with varying β